

Rekonstruktion von Softwarearchitekturen durch dynamische Analyse

Prof. Dr. Wilhelm Hasselbring

mit Beiträgen von
André van Hoorn, Nils Ehmke, Sören Frey, Reiner Jung,
Holger Knoche (b+m), Matthias Rohr, Jan Waller
und weiteren

Software Engineering Group, Univ. Kiel
<http://se.informatik.uni-kiel.de/>

GI-Fachtagung Architekturen 2012

Paderborn, 2. Juli 2012



Unterschiedliche Zielsetzungen und Werkzeuge, einige Beispiele:

Code Metriken JavaNCSS
Programmierrichtlinien Checkstyle
Fehlermuster FindBugs
Architekturanalyse Sotograph
Klonerkennung Bauhaus

Statische Analyse kann bei Vorliegen des Quellcodes direkt angewendet werden um Qualitätseigenschaften zu überprüfen.

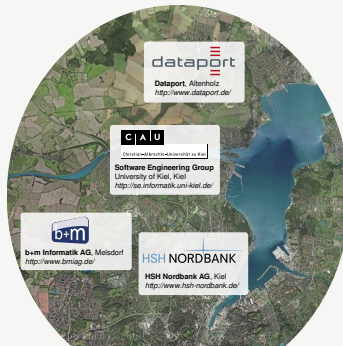
- Statische Analyse kann Einblicke in die Struktur eines Softwaresystems liefern,
 - das **tatsächliche** Laufzeitverhalten kann jedoch nur durch eine dynamische Analyse ermittelt werden.
- Profiling (in der Entwicklung) ermöglicht eine dynamische Analyse,
 - üblicherweise jedoch nicht unter realen Lastbedingungen.
- Monitoring (im Betrieb) ermöglicht die kontinuierliche dynamische Analyse
 - des tatsächlichen Laufzeitverhaltens unter realen Lastbedingungen.
- Ein Verständnis der Bedürfnisse des **Betriebs** von Softwaresystemen wird benötigt, um
 - im **Konstruktionsprozess** die richtigen Aktivitäten der
 - zum **Monitoring** für den jeweiligen **Einsatzzweck** benötigten Instrumentierung der Software durchzuführen.

Historisch gewachsene Architekturen



KoSSE-Projekt DynaMod

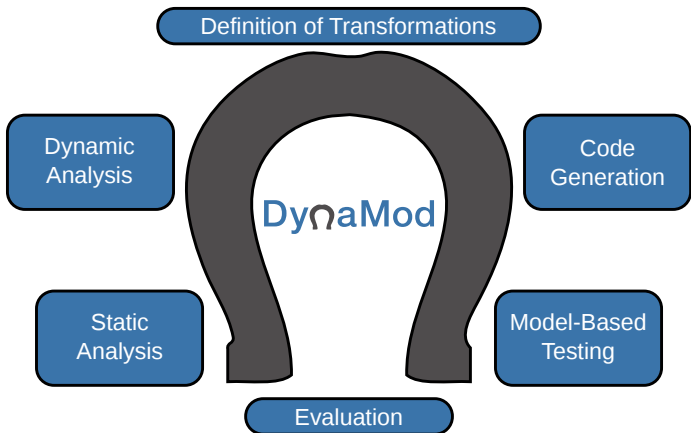
Dynamische Analyse zur modellgetriebenen Modernisierung



GEFÖRDERT VOM



Bundesministerium
für Bildung
und Forschung



- 1 Dynamische Analyse und Modernisierung
- 2 Kieker Monitoring & Analyse Framework
 - Nutzungsszenario
 - Architektur
 - Pipe-and-Filter Konfiguration
- 3 Reverse Engineering Beispiele
 - Reverse Engineering von Java EE
 - Reverse Engineering von C#
 - Reverse Engineering von COBOL
 - Reverse Engineering von Visual Basic 6
 - Calling Networks
 - 3D-Visualisierung
- 4 Zusammenfassung und Ausblick

Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis

www.kieker-monitoring.net

Continuous Application Performance Monitoring for Online and Offline Analysis

- Static analysis is not sufficient to study the internal behavior of software systems comprehensively
- Continuous monitoring allows to gather a system's actual runtime behavior resulting from production usage profiles
- The obtained monitoring data can, for instance, be used for
 - Performance evaluation (e.g., bottleneck detection)
 - Simulation (workload, measurement, logging, and analysis)
 - Self-adaptation control (e.g., capacity management)
 - Software maintenance, reverse engineering, modernization
 - Application-level failure detection and diagnosis
 - Service-level management

Instrumentation and Monitoring Overhead

- Includes probes for collecting timing, control flow, and resource utilization
- Support for various Java instrumentation methods, e.g., AspectJ, middleware interception, Servlet filters
- Micro-benchmarks revealed low overhead
- Each activation adds constant overhead (linear scaling)

Framework Characteristics

- Modular, flexible, and extensible architecture
- Extensible probes, readers, writers, records, and plugins
- Integrated monitoring record type model for monitoring and analysis
- Allows to log, reconstruct, analyze, and visualize distributed traces
- Designed for continuous operation in multi-user systems
- Evaluated in lab experiments and industrial case studies (since 2006)
- Kieker is open-source software (Apache License, V.2.0)

Pipe-and-Filter Configuration for Analysis/Visualization

Invited Tool Demo @ ICPE 2012

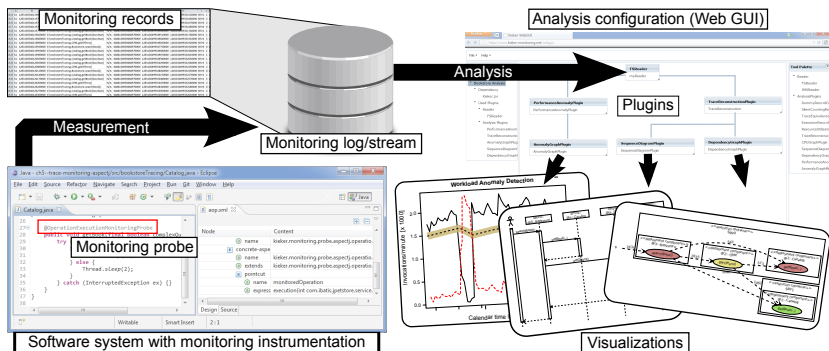
A. van Hoorn, J. Walker, and W. Hasselbring
 Kieker: A Framework for Application Performance Monitoring and Dynamic Software Analysis
 Proc. 3rd ACM/SPEC Int. Conf. Perform. Eng. (ICPE'12), ACM, 2012

Current Activities/Coming Soon

- Monitoring adapters for .NET, VB6, COBOL etc.
- Model-driven instrumentation & analysis
- Web-based graphical user interface
- Plugins for analysis of concurrent behavior

Software Engineering Group
 Department of Computer Science
 University of Kiel, Germany

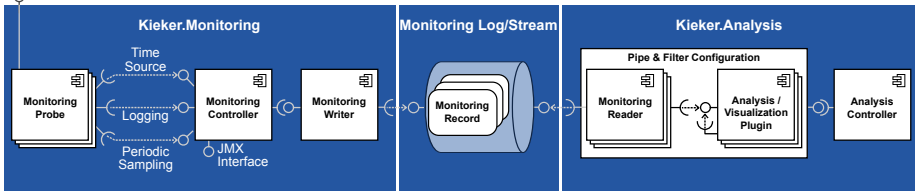
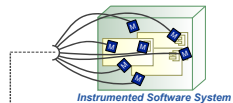
Kieker is distributed as part of SPECeB RCU's repository of peer-reviewed tools for quantitative system evaluation and analysis



Kieker Monitoring & Analyse Framework



Kieker Monitoring & Analyse Framework ▷ Architektur



Kieker Monitoring & Analyse Framework

Wesentliche Eigenschaften [van Hoorn et al. 2009; 2012]

Kieker Monitoring & Analyse Framework ▷ Architektur



- Flexible, modular erweiterbare Architektur (Messsonden, Log-Datenstrukturen, Analyse Plug-Ins, etc.)
- Verteiltes Tracing (Spring, SOAP, etc.)
- Niedriger Overhead (entworfen für den kontinuierlichen Betrieb)
- Evaluiert im Labor und in industriellen Umgebungen



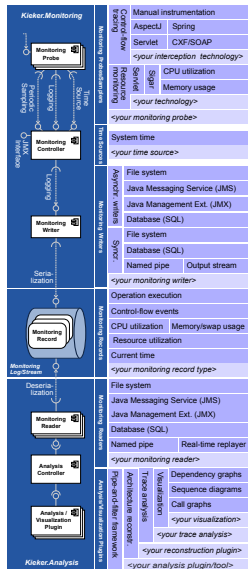
Kieker ist Open-Source Software (Apache License, V. 2.0)

<http://kieker-monitoring.net>

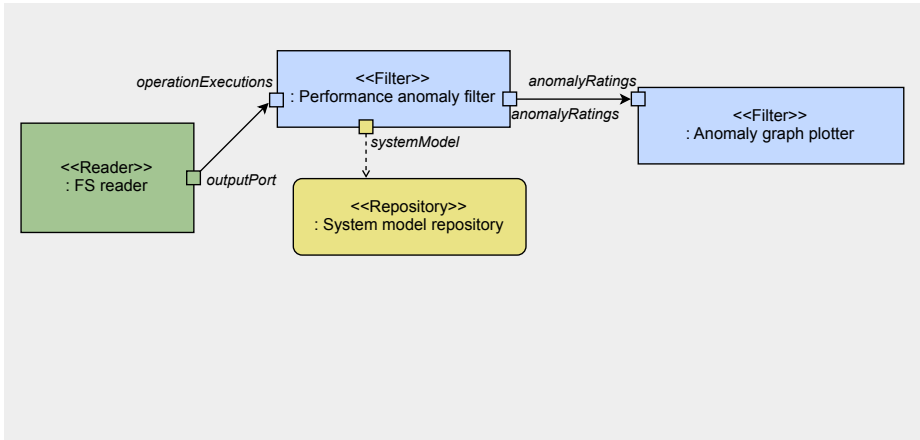
Empfohlenes Tool der SPEC Research Group

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>



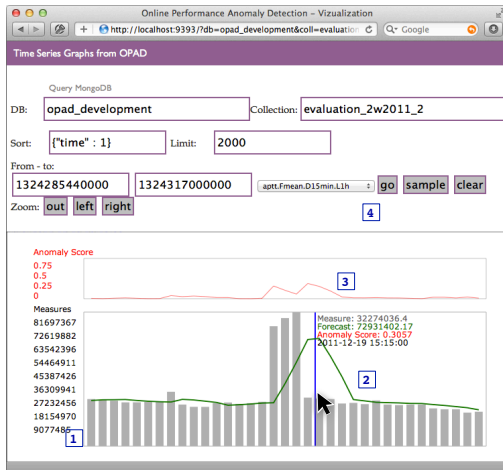
Beispiel Pipe-and-Filter Konfiguration



Performance-Anomalieerkennung

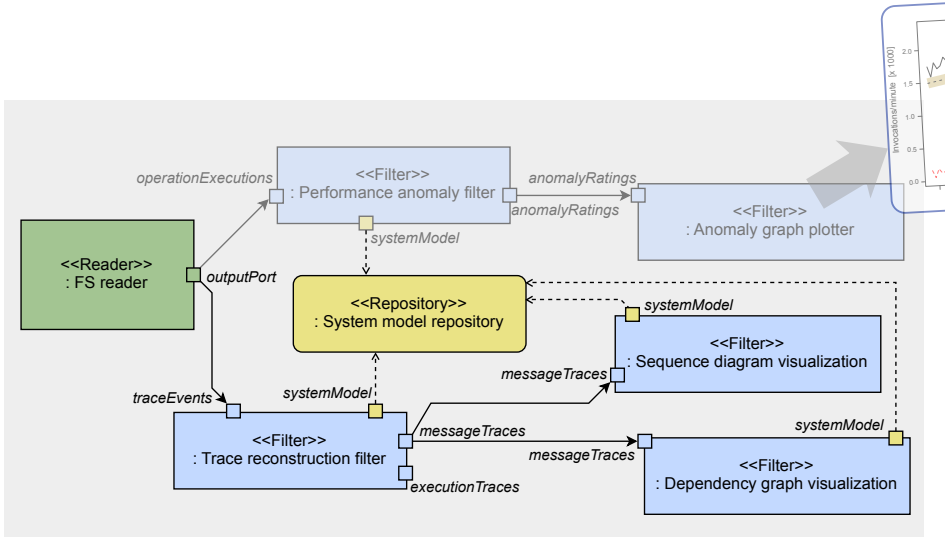
Ein typisches Beispiel für die Nutzung eines Monitoring-Frameworks

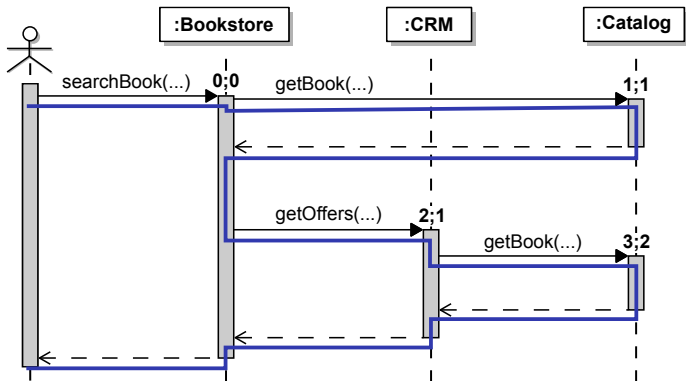
Kieker Monitoring & Analyse Framework ▷ Pipe-and-Filter Konfiguration




Entwickelt und eingesetzt bei der Xing AG [Bielefeld 2012].

Beispiel Pipe-and-Filter Konfiguration





Legend:

→ = call message  = trace $i;j$ = execution with eoi i and ess j
← - = return message

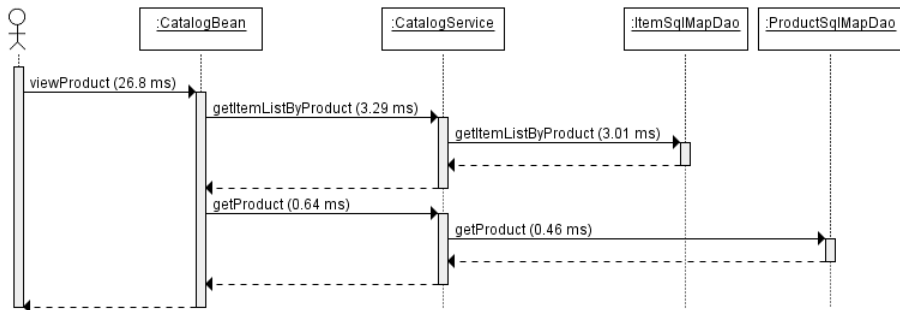
Execution order index (eoi) i : i -th started execution in a trace

Execution stack size (ess) j : execution started at stack depth j

Generiertes Sequenzdiagramm

Visualisierungsbeispiel für die dynamische Analyse

Kieker Monitoring & Analyse Framework ▷ Pipe-and-Filter Konfiguration



Web GUI zur Konfiguration der Analyse

(in Arbeit)

Kieker Monitoring & Analyse Framework ▷ Pipe-and-Filter Konfiguration



The screenshot displays the Kieker Web GUI interface for configuring an analysis. The main workspace shows a pipe-and-filter diagram with the following components and connections:

- FSReader** (Remove) receives `monitoringRecords` and outputs `executions`.
- TraceReconstructionFilter** (Remove) receives `executions` and outputs `messageTraces`.
- SystemModelRepository** (Remove) receives `messageTraces` and outputs `systemModelRepository`.
- KDMExtractorFilter** (Remove) receives `messageTraces` and outputs `msgTraces`.
- KDMRepository** (Remove) receives `msgTraces` and outputs `kdmRepository`.

The **Available Plugins** panel on the right lists the following categories:

- Reader:** DfReader, FSReader, JMSReader, JMSReader, SysReader, FSReaderResultmg
- Filter:**
- Repositories:** KDMRepository, SystemModelRepository

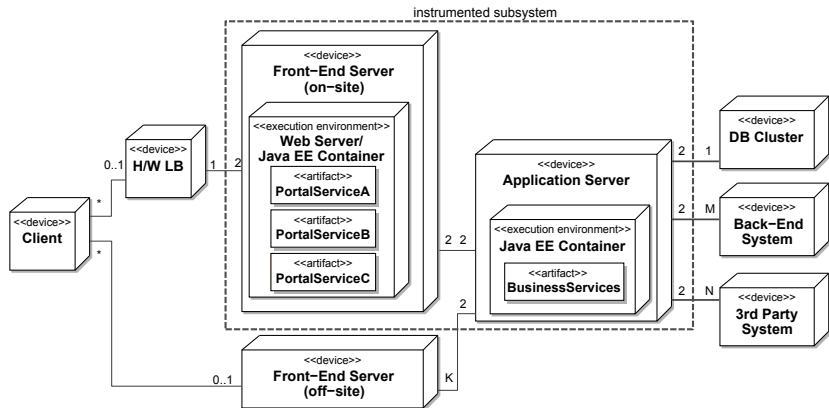
The **Properties** panel at the bottom shows the configuration for the `KDMExtractorFilter`:

Key	Value	Options
Name	KDMExtractorFilter	✓
outputFile	result.xml	✓

- 1 Dynamische Analyse und Modernisierung
- 2 Kieker Monitoring & Analyse Framework
 - Nutzungsszenario
 - Architektur
 - Pipe-and-Filter Konfiguration
- 3 **Reverse Engineering Beispiele**
 - Reverse Engineering von Java EE
 - Reverse Engineering von C#
 - Reverse Engineering von COBOL
 - Reverse Engineering von Visual Basic 6
 - Calling Networks
 - 3D-Visualisierung
- 4 Zusammenfassung und Ausblick

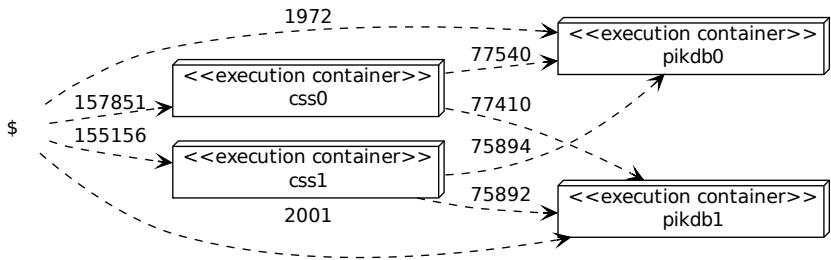


Verteiltes Enterprise Java System



Servlet, Spring und CXF/SOAP Messsonden

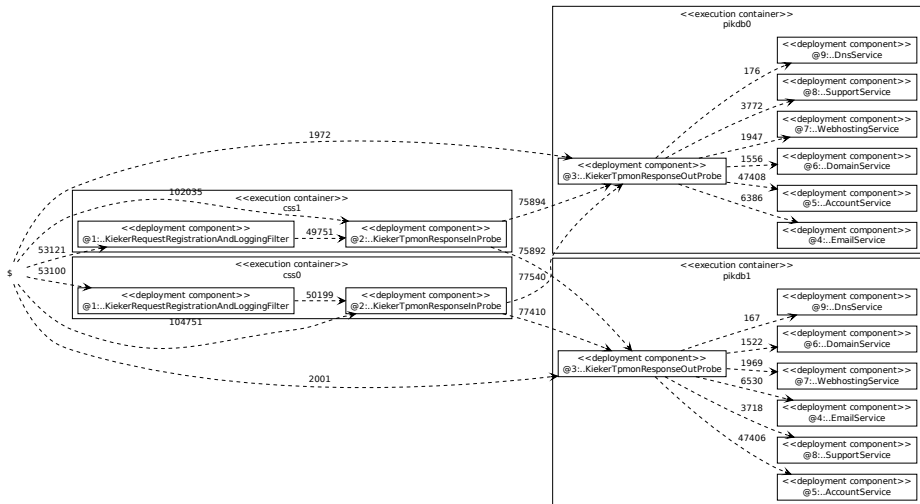




Komponenten-Abhängigkeitsgraph

Systemarchitekturebene

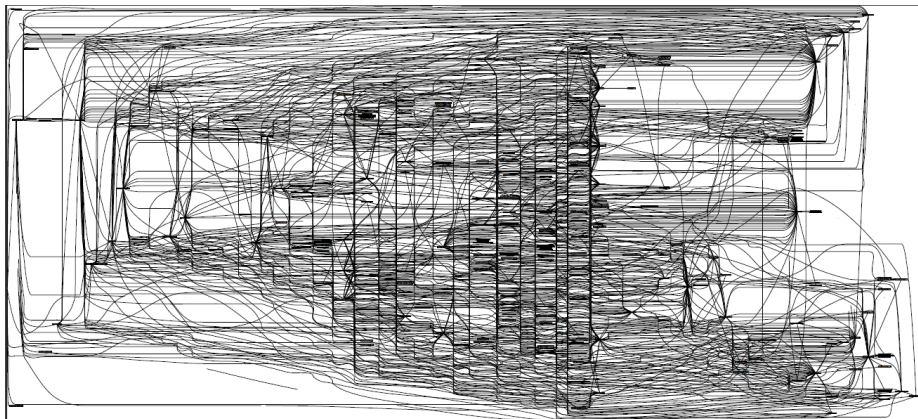
Reverse Engineering Beispiele ▸ Reverse Engineering von Java EE



Reverse Engineering von C#

Vollständige Test-Suite für Nordics Analytics [Magendanz 2011]

Reverse Engineering Beispiele ▸ Reverse Engineering von C#



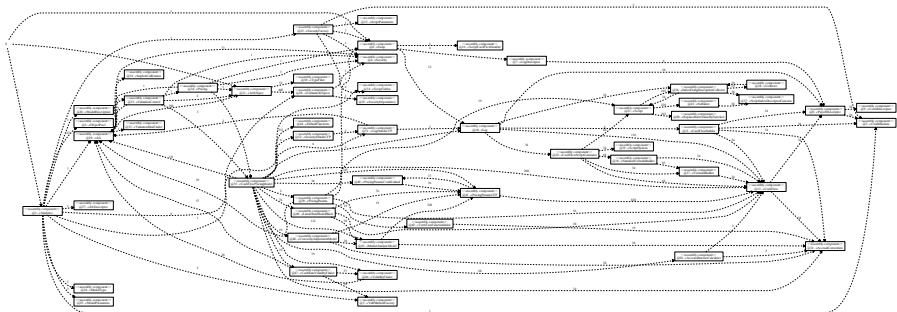
Fallstudie bei der HSH Nordbank AG.



Nach Auswahl eines Anwendungsfalles

Reverse Engineering von C#

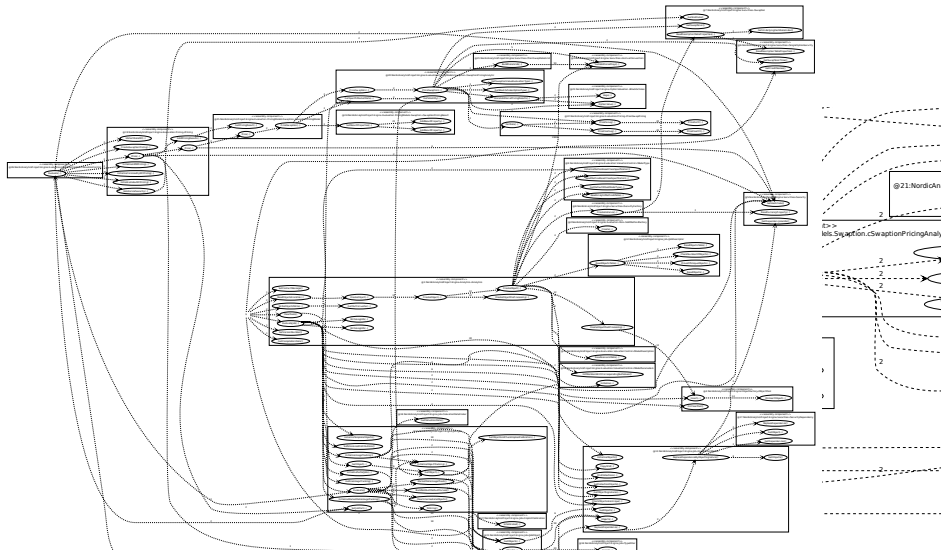
Reverse Engineering Beispiele ▸ Reverse Engineering von C#



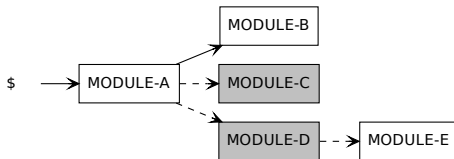
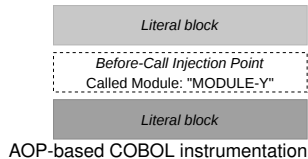
Zoom auf die Operationsebene

Reverse Engineering von C#

Reverse Engineering Beispiele ▸ Reverse Engineering von C#



```
IDENTIFICATION DIVISION.  
PROGRAM-ID. MODULE-X.  
  
PROCEDURE DIVISION.  
  
CALL "MODULE-Y".  
GOBACK.
```



Module-level call dependency graph with assumed dependencies

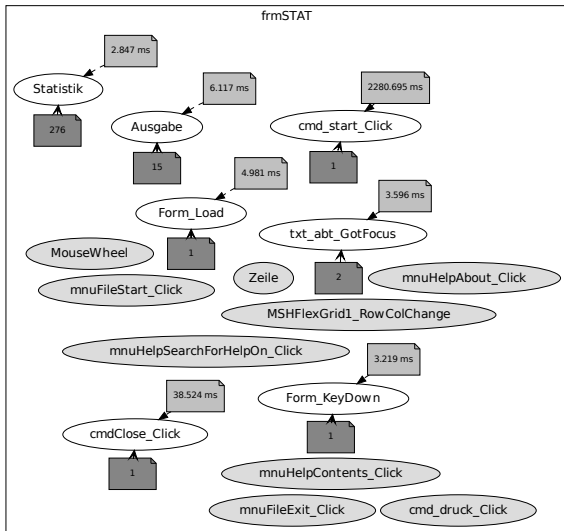
Fallstudie der b+m Informatik AG (Kieker-Erweiterungen durch Holger Knoche):

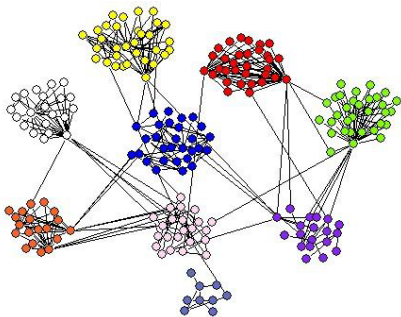
- 1 Industrieller Kontext mit mehr als 1.000 COBOL Modulen.
- 2 140.351 Messpunkte mit 14 verschiedenen Arten von Messsonden.

Erkennung nicht genutzter Funktionen

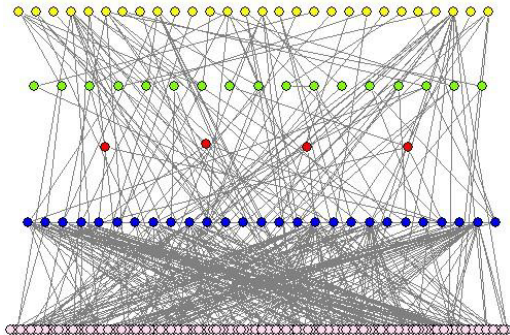
Reverse Engineering von Visual Basic 6

Reverse Engineering Beispiele > Reverse Engineering von Visual Basic 6





Community structure



Layered structure (hierarchy)

Xi'an Jiaotong University, Shaanxi [Zheng et al. 2011]

3D-Visualisierung mit DyVis [Wulf 2010]

Visualisierungsbeispiel

Reverse Engineering Beispiele > 3D-Visualisierung



The screenshot shows the DyVis application window. The main area displays a 3D visualization of a software architecture with blue blocks representing classes and green lines representing relationships. The interface includes a menu bar (File, Edit, Help), a toolbar with navigation buttons (< 1 time unit, < 1 event, Play, 1 event >, 1 time unit >), and a right-hand pane showing a hierarchical tree of classes and methods. The tree is expanded to show the following structure:

- global
- DispatcherServlet
 - doGet
 - process
 - addItemToCart
 - addItemToStock
 - getItem
- DispatcherServlet
 - doGet
 - process
 - viewCategory
 - getProductListByCate
 - getCategory
- DispatcherServlet
- DispatcherServlet
 - doGet
 - process
 - viewCategory
 - getProductListByCate
 - getCategory
- DispatcherServlet
 - doGet
 - process
 - viewCategory
 - getProductListByCate
 - getCategory
- DispatcherServlet
 - doGet
 - process
 - viewProduct
- DispatcherServlet
- DispatcherServlet

At the bottom, there is a table with two columns, A and B, showing call stack information:

A	B
Name	addItemToCart
Filename	nullCartBean
Line	54
caller class	DispatcherServlet
callee class	CartBean
Trace Identifier	3194037298224168972
left	

Available trace elements: 88/101

Zusammenfassung

- Das Monitoring zur dynamischen Analyse liefert wichtige Daten über das **tatsächliche** Verhalten eines Systems.
 - Kombiniert mit statischer Analyse.
- Die Monitoring-Daten können zur **Rekonstruktion** der Softwarearchitekturen genutzt werden.
- Für die Architektur-Rekonstruktion sind **flexible Anpassungsmöglichkeiten** erforderlich.
- In der Entwicklung sind spezifische **Entwurfsentscheidungen** für das Monitoring erforderlich.

Aktuelle Arbeiten u.a.:

- Web-basierte Konfigurations-Oberfläche & Cockpit
- Model-driven instrumentation & analysis
- Einsatz für Workflow-Monitoring und zum Profiling eingebetteter Systeme

- T. C. Bielefeld. Online performance anomaly detection for large-scale software systems. Diploma thesis, University of Kiel, Germany, Mar. 2012.
- H. Knoche, A. van Hoorn, W. Goerigk, and W. Hasselbring. Automated source-level instrumentation for dynamic dependency analysis of COBOL systems. In *Proceedings of the 14. Workshop Software-Reengineering (WSR '12)*, pages 33–34, May 2012.
- F. Magendanz. Dynamic analysis of .NET applications for architecture-based model extraction and test generation, Oct. 2011.
- A. van Hoorn, M. Rohr, W. Hasselbring, J. Waller, J. Ehlers, S. Frey, and D. Kieselhorst. Continuous monitoring of software services: Design and application of the Kieker framework. Technical Report TR-0921, Department of Computer Science, University of Kiel, Germany, Nov. 2009. URL http://www.informatik.uni-kiel.de/uploads/tx_publication/vanhoorn_tr0921.pdf.
- A. van Hoorn, J. Waller, and W. 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 2012)*, pages 247–248. ACM, Apr. 2012. ISBN 978-1-4503-1202-8.
- C. Wulf. Runtime visualization of static and dynamic architectural views of a software system to identify performance problems. B.Sc. Thesis, University of Kiel, Germany, 2010.
- Q. Zheng, Z. Ou, L. Liu, and T. Liu. A novel method on software structure evaluation. In *Proceedings of the 2nd IEEE International Conference on Software Engineering and Service (IEEE ICSESS 2011)*, pages 251–254. IEEE, July 2011. doi: 10.1109/ICSESS.2011.5982301.