Christian-Albrechts-Universität zu Kiel

Architecture Reconstruction via Dynamic Analysis

Wilhelm Hasselbring and André van Hoorn

Software Engineering Group, University of Kiel
`http://se.informatik.uni-kiel.de/`

June 29, 2011 @ SEACON 2011, Hamburg

“**Dynamic analysis**, or the *analysis of data gathered from a running program*, has the potential to **provide an accurate picture of a software** system because it exposes the **system's actual behavior.**

This **picture can range from class-level details up to high-level architectural views** [...].”

(Cornelissen et al. 2009)

Dynamic Analysis of Software Systems (cont'd)

"Among the **benefits over static analysis** are the availability of runtime information and, in the context of object-oriented software, the *exposure of object identities* and the actual resolution of late binding.

A **drawback** is that dynamic analysis can only provide a **partial picture of the system**, i.e., the results obtained are valid for the **scenarios that were exercised during the analysis**."

(Cornelissen et al. 2009)

Dynamic Analysis of Software Systems (cont'd)

How to Gather Runtime Data from Executing Systems? — Instrumentation

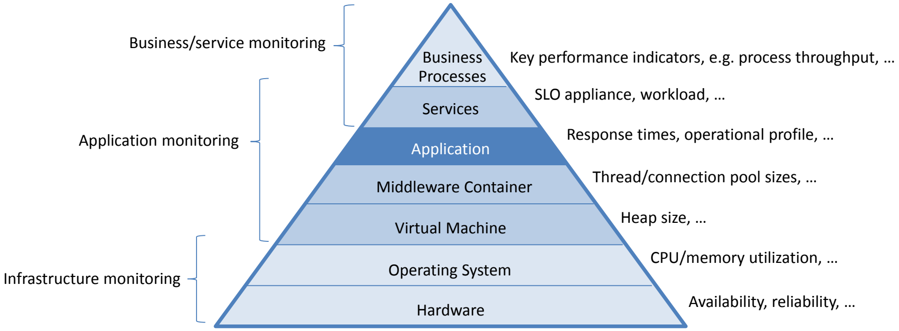
- Profiling — employed in development environments; considerable performance overhead
- Monitoring — employed in production environments; captures real usage profile

Use Cases — Online & Offline Analysis

The obtained monitoring data can, for instance, be used for

- **Performance evaluation** (e.g., bottleneck detection)
- **(Self-)adaptation control** (e.g., capacity management)
- Application-level **failure detection and diagnosis**
- Simulation/Testing (workload, measurement, logging, and analysis)
- Software maintenance, **reverse engineering**, modernization
- Service-level management

Continuous Monitoring of Software Systems


Business/service monitoring
Application monitoring
Infrastructure monitoring
Business Processes
Services
Application
Middleware Container
Virtual Machine
Operating System
Hardware
Key performance indicators, e.g. process throughput, …
SLO appliance, workload, …
Response times, operational profile, …
Thread/connection pool sizes, …
Heap size, …
CPU/memory utilization, …
Availability, reliability, …

Application-Level Monitoring in Practice . . . !?

— At Facebook —

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,

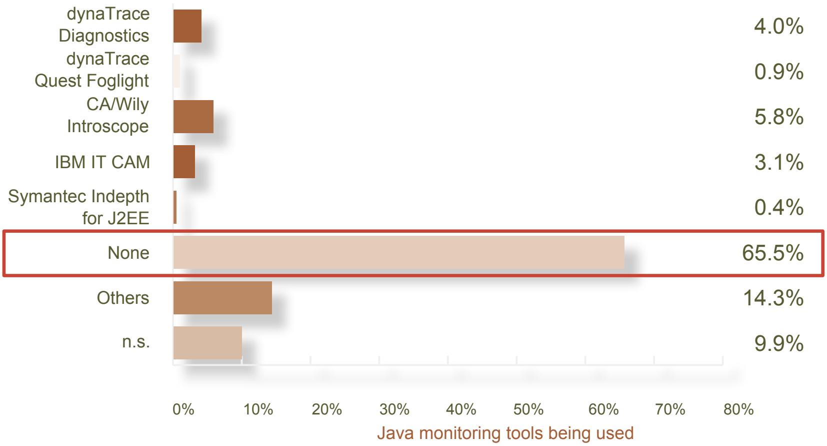
it’s also **application level statistics** to understand why things are happening.

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

Application-Level Monitoring in Practice . . . !?

— In General —



"Java monitoring largely unknown."
[codecentric GmbH 2009]

Continuous Monitoring — Design Decisions
[van Hoorn et al. 2009a]

Model-Driven Instrumentation & Analysis

Selection of Monitoring Probes
Depends on analysis goals

Number and Position of Monitoring Points
Trade-off between performance overhead and information quality

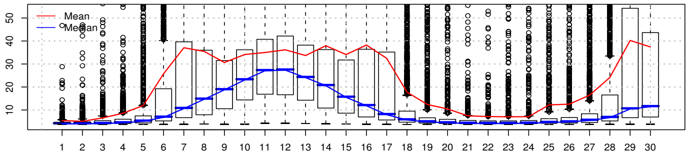
Explicit Consideration of the Induced Monitoring Overhead
During continuous operation: only small, constant overhead acceptable

Physical Location of the Monitoring Log/Stream
E.g., relational database, file system, messaging queue

Intrusiveness of Instrumentation
Concern: mixing monitoring logic with business functionality; $\rightarrow$ AOP

Agenda

Monitoring & Analysis of Software Behavior



Visualization of results

Instrumentation

```
public static void searchBook() {
    long tin = System.currentTimeMillis();
    Catalog.getBook(false);
    long tout = System.currentTimeMillis();
    System.out.printf("Catalog;getBook;%s;%s", tin, tout);
}
```

Program execution

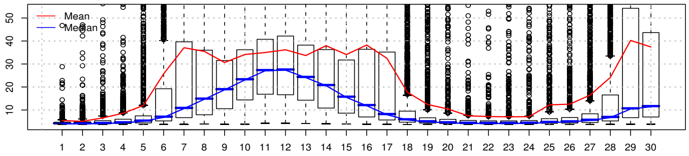
```
Catalog;getBook;1273333822196;1273333822202
Catalog;getBook;1273333823197;1273333823199
Catalog;getBook;1273333824198;1273333824200
Catalog;getBook;1273333825199;1273333825201
Catalog;getBook;1273333826200;1273333826202
```

Monitoring log

Statistical analysis

Example: Monitoring, analysis, and visualization of operation response times

Monitoring & Analysis of Software Behavior



Instrumentation

```
public static void searchBook() {
    long tin = System.currentTimeMillis();
    Catalog.getBook(false);
    long tout = System.currentTimeMillis();
    System.out.printf("Catalog;getBook;%s;%s", tin, tout);
}
```

Visualization of results

Statistical analysis

Program execution

```
Catalog;getBook;1273333822196;1273333822202
Catalog;getBook;1273333823197;1273333823199
Catalog;getBook;1273333824198;1273333824200
Catalog;getBook;1273333825199;1273333825201
Catalog;getBook;1273333826200;1273333826202
```

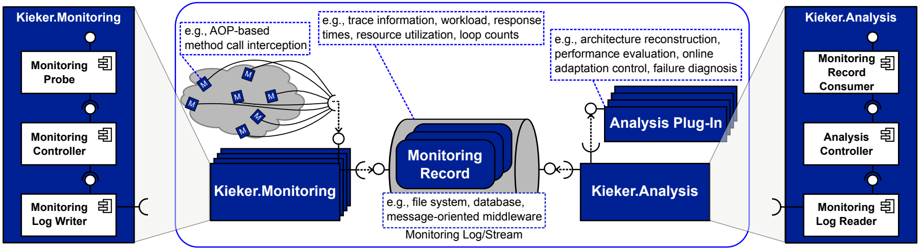
Monitoring log

Issues (e.g.)

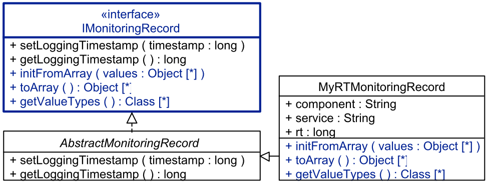
Maintainability, changing log format, additional/more complex data structures, exception handling, alternative data sinks (e.g., database)

Kieker Monitoring & Analysis Framework

Kieker
http://kieker.sourceforge.net


Kieker.Monitoring
Monitoring Probe
Monitoring Controller
Monitoring Log Writer
e.g., AOP-based method call interception
e.g., trace information, workload, response times, resource utilization, loop counts
e.g., architecture reconstruction, performance evaluation, online adaptation control, failure diagnosis
M
Kieker.Monitoring
Monitoring Record
e.g., file system, database, message-oriented middleware
Monitoring Log/Stream
Analysis Plug-In
Kieker.Analysis
Kieker.Analysis
Monitoring Record Consumer
Analysis Controller
Monitoring Log Reader

Examples: Monitoring Record, Probe, Plugin


«interface»
IMonitoringRecord
+ setLoggingTimestamp (timestamp : long)
+ getLoggingTimestamp ( ) : long
+ initFromArray (values : Object [*])
+ toArray ( ) : Object [*]
+ getValueTypes ( ) : Class [*]
AbstractMonitoringRecord
+ setLoggingTimestamp (timestamp : long)
+ getLoggingTimestamp ( ) : long
MyRTMonitoringRecord
+ component : String
+ service : String
+ rt : long
+ initFromArray (values : Object [*])
+ toArray ( ) : Object [*]
+ getValueTypes ( ) : Class [*]

Custom Monitoring Record: MyRTMonitoringRecord

```
@Around(value = "execution(@mySimpleKiekerExample.annotation.MyRTProbe * *.
  public Object probe(ProceedingJoinPoint j) throws Throwable {
    MyRTMonitoringRecord record = new MyRTMonitoringRecord();
    record.component = j.getSignature().getDeclaringTypeName();
    record.service = j.getSignature().getName();
    Object retval;
...
    } finally {
      record.rt = CTRL.getTime() - tin;
      CTRL.newMonitoringRecord(record);
    }
    return retval;
```

(AOP-based) Monitoring Probe

```
  public boolean newMonitoringRecord(IMonitoringRecord r) {
    MyRTMonitoringRecord rtRec = (MyRTMonitoringRecord) r;
    if (rtRec.rt > this.rtSLO) {
      log.info("rtRec.rt >this.rtSLO: " + rtRec.rt + ">" + this.rtSLO);
    } else {
      log.info("rtRec.rt <=this.rtSLO: " + rtRec.rt + "<=" + this.rtSLO)
    }
    return true;
  }
```

Analysis plug-in: Response time evaluation

Characteristics & Features

Characteristics

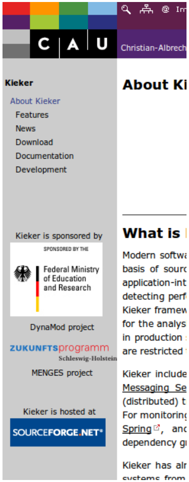
- Flexible architecture (custom *probes*, *readers*, *writers*, *analysis plug-ins*)
- Integrated & extensible *record* type model for monitoring & analysis
- Enables offline and online analysis/visualization
- Low overhead (designed for continuous operation in multi-user systems)
- Evaluated in industry case studies

Features

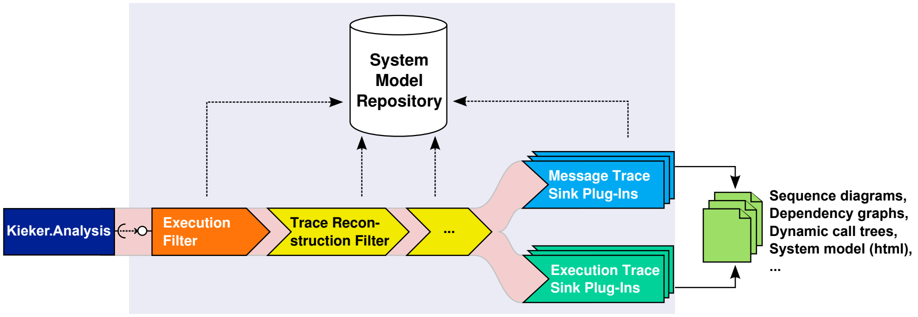
- Readers/Writers: File system, database, messaging (JMS), named pipe
- Probes: Operation executions (AOP-based); cpu, memory/swap (Sigar API)
- Logging, reconstruction, analysis/visualization of (distributed) traces
- Visualizations: Dependency & sequence diagrams, call trees, system model
- Log replayer (also in real-time)
- Moreover: Periodic sampling, custom time sources, control via JMX etc.

Kieker is Open Source Software— Current Version: 1.3

- Web site: `http://kieker.sourceforge.net`


Christian-Albrech
Kieker
About Kieker
Features
News
Download
Documentation
Development
About Ki
What is
Modern softwa
basis of sourc
application-int
detecting perf
Kieker framew
for the analys
in production
are restricted
Kieker include
Messaging Se
(distributed) t
For monitoring
Spring, and
dependency g
Kieker has alr
Kieker is sponsored by
Federal Ministry of Education and Research
DynaMod project
ZUKUNFTSprogramm Schleswig-Holstein
MENGES project
Kieker is hosted at
SOURCEFORGE.NET

Kieker.TraceAnalysis Tool

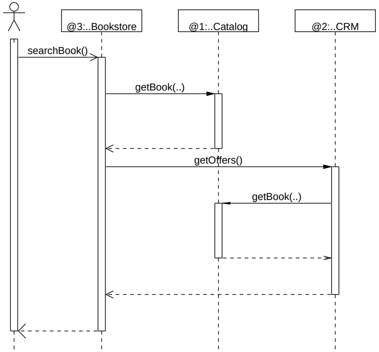


Kieker.TraceAnalysis Tool

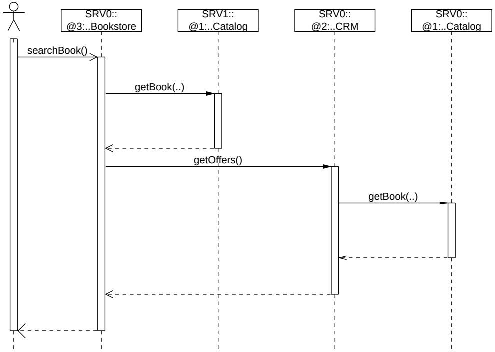
Sequence Diagrams

Kieker.TraceAnalysis Tool (cont'd)

1. **Sequence diagrams**
2. Dynamic call trees
3. Hierarchical calling dependency graphs
4. System model



(a) Assembly-level view

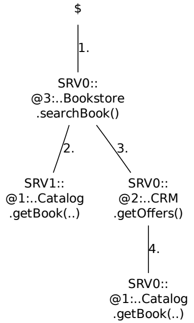


(b) Deployment-level view

Dynamic Call Trees

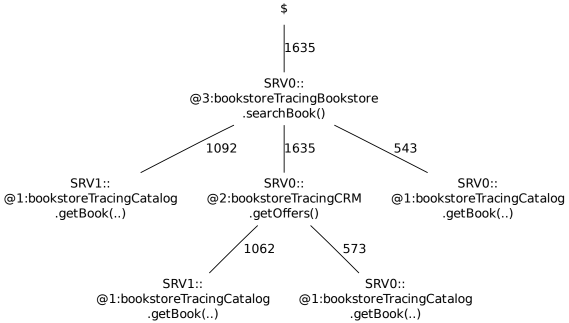
Kieker.TraceAnalysis Tool (cont'd)

1. Sequence diagrams
2. **Dynamic call trees**
3. Hierarchical calling dependency graphs
4. System model



$
1.
SRV0:: @3:..Bookstore .searchBook()
2.
3.
SRV1:: @1:..Catalog .getBook(..)
SRV0:: @2:..CRM .getOffers()
4.
SRV0:: @1:..Catalog .getBook(..)

(a) Dynamic call tree **(single trace)**



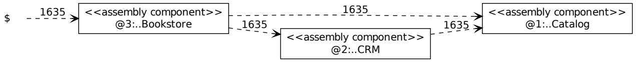
$
1635
SRV0:: @3:bookstoreTracingBookstore .searchBook()
1092
1635
543
SRV1:: @1:bookstoreTracingCatalog .getBook(..)
SRV0:: @2:bookstoreTracingCRM .getOffers()
SRV0:: @1:bookstoreTracingCatalog .getBook(..)
1062
573
SRV1:: @1:bookstoreTracingCatalog .getBook(..)
SRV0:: @1:bookstoreTracingCatalog .getBook(..)

(b) **Aggregated** deployment-level call tree

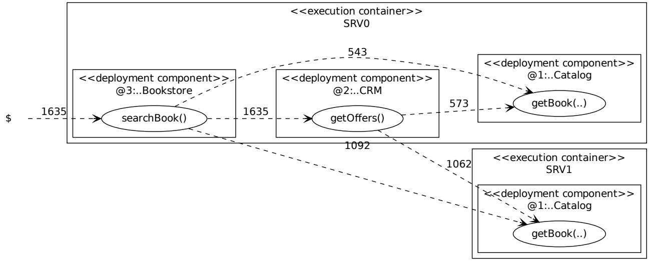
Hierarchical Calling Dependency Graphs

Kieker.TraceAnalysis Tool (cont'd)

1. Sequence diagrams
2. Dynamic call trees
3. **Hierarchical calling dependency graphs**
4. System model



(a) Assembly-level **component** dependency graph

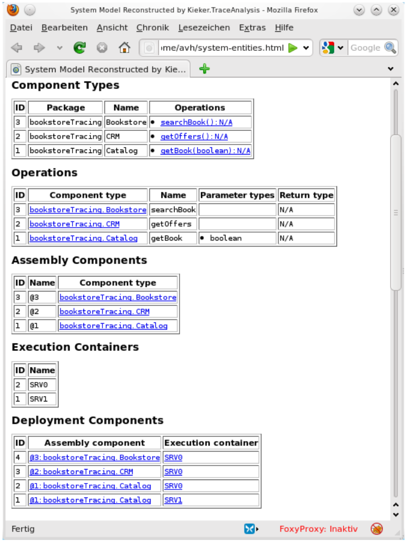


(b) Deployment-level **operation** dependency graph

System Model (HTML Representation)

Kieker.TraceAnalysis Tool (cont'd)

1. Sequence diagrams
2. Dynamic call trees
3. Hierarchical calling dependency graphs
4. **System model** (here: HTML representation)


System Model Reconstructed by Kieker.TraceAnalysis - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
ıme/avh/system-entities.html
Google
System Model Reconstructed by Kie...
Fertig
FoxyProxy: Inaktiv

Component Types

ID	Package	Name	Operations
3	bookstoreTracing	Bookstore	• searchBook():N/A
2	bookstoreTracing	CRM	• getOffers():N/A
1	bookstoreTracing	Catalog	• getBook(boolean):N/A

Operations

ID	Component type	Name	Parameter types	Return type
3	bookstoreTracing.Bookstore	searchBook		N/A
2	bookstoreTracing.CRM	getOffers		N/A
1	bookstoreTracing.Catalog	getBook	• boolean	N/A

Assembly Components

ID	Name	Component type
3	@3	bookstoreTracing.Bookstore
2	@2	bookstoreTracing.CRM
1	@1	bookstoreTracing.Catalog

Execution Containers

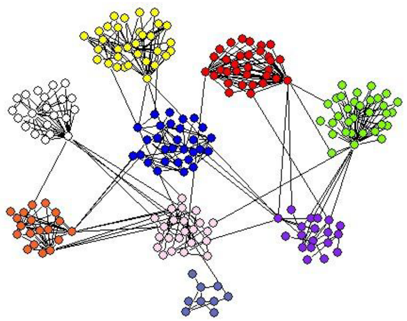
ID	Name
2	SRV0
1	SRV1

Deployment Components

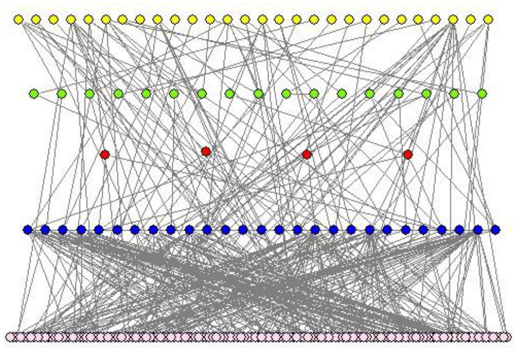
ID	Assembly component	Execution container
4	@3:bookstoreTracing.Bookstore	SRV0
3	@2:bookstoreTracing.CRM	SRV0
2	@1:bookstoreTracing.Catalog	SRV0
1	@1:bookstoreTracing.Catalog	SRV1

Calling Networks

Clustering based on Calling (Frequency) Relationships [Zheng et al. 2011]



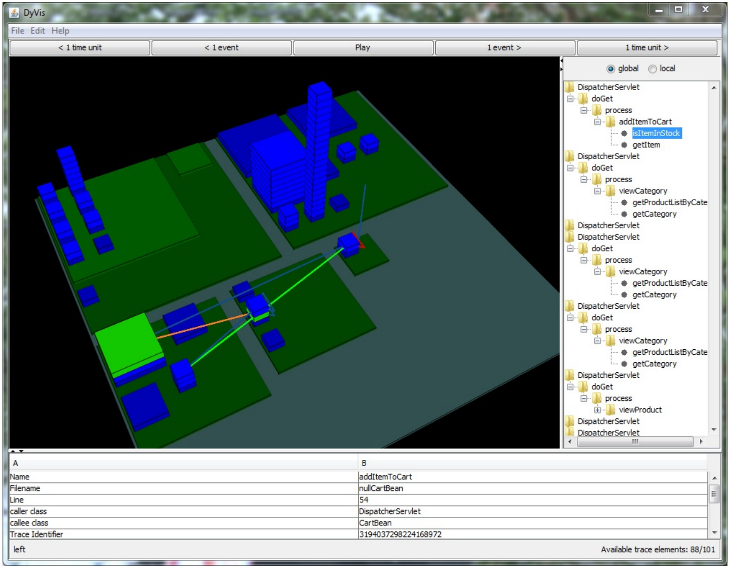
Community structure



Layered structure (hierarchy)

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

3D Visualization with DyVis
[Wulf 2010]



```
DispatcherServlet
  doGet
    process
      addItemToCart
        isItemInStock
        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
```

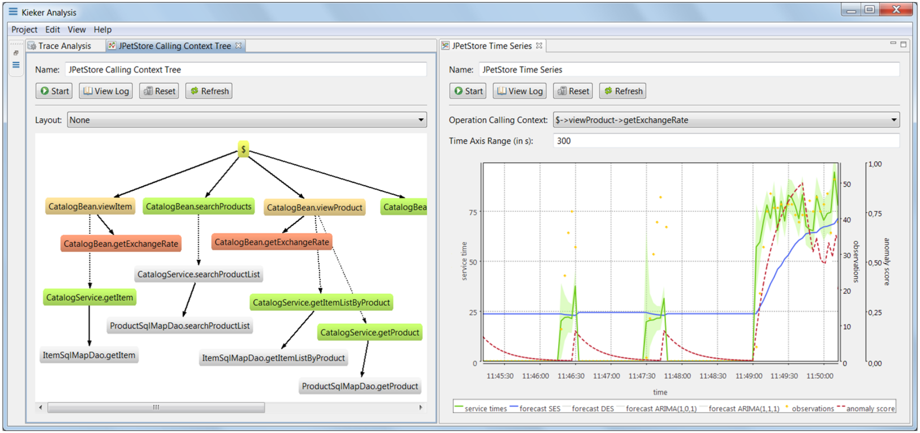
A	B
Name	addItemToCart
Filename	nullCartBean
Line	54
caller class	DispatcherServlet
callee class	CartBean
Trace Identifier	3194037298224168972

left

Available trace elements: 88/101

Anomaly Detection and Diagnosis

[Ehlers and Hasselbring 2011]



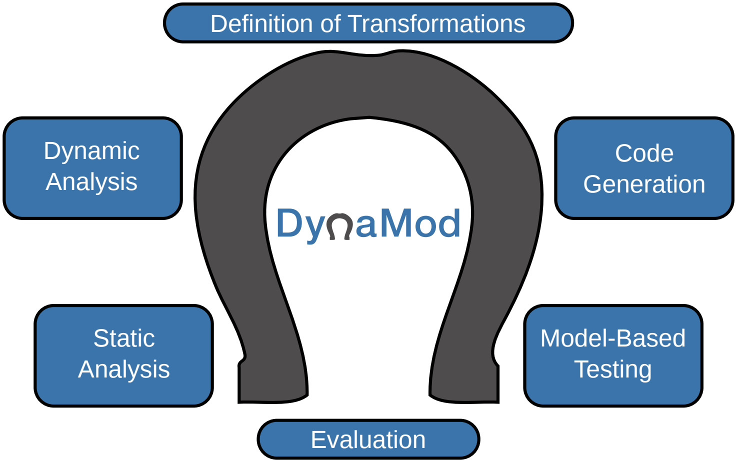
Agenda

Christian-Albrechts-Universität zu Kiel

Context — DynaMod Project

DynaMod

Dynamic Analysis for
Model-Driven Software Modernization



Context — DynaMod Project (cont'd)

DynaMod

Dynamic Analysis for
Model-Driven Software Modernization

Project Consortium:

1. **b+m Informatik AG**
   (Development partner, consortium leader)
2. **Software Engineering Group, Univ. Kiel**
   (Research partner)
3. **Dataport**
   (Associated partner)
4. **HSH Nordbank AG**
   (Associated partner)

Funding:

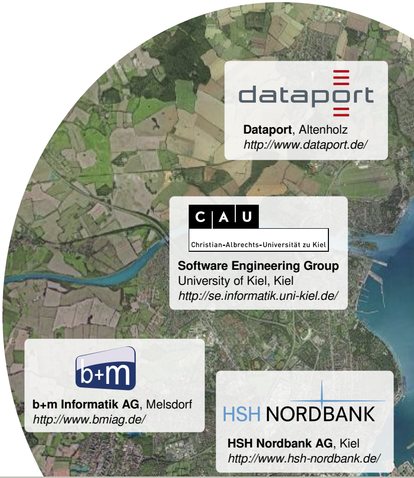
- BMBF "KMU-innovativ"
- 2 years (01/11–12/12)

SPONSORED BY THE
Federal Ministry
of Education
and Research

Under grant no. 01IS10051

Read More:

- `http://kosse-sh.de/dynamod/`
- MDSM @ CSCMR 2011 [van Hoorn et al. 2011a]



dataport
Dataport, Altenholz
http://www.dataport.de/

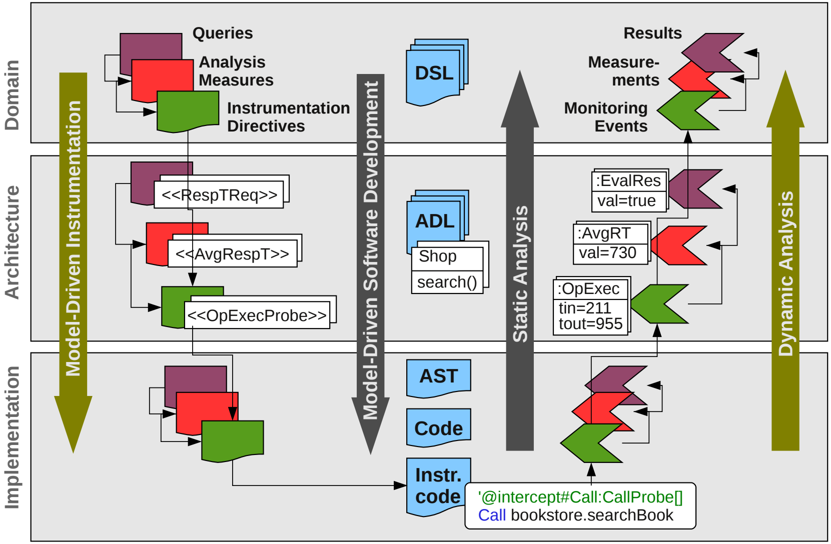
C | A | U
Christian-Albrechts-Universität zu Kiel
Software Engineering Group
University of Kiel, Kiel
http://se.informatik.uni-kiel.de/

b+m
b+m Informatik AG, Melsdorf
http://www.bmiag.de/

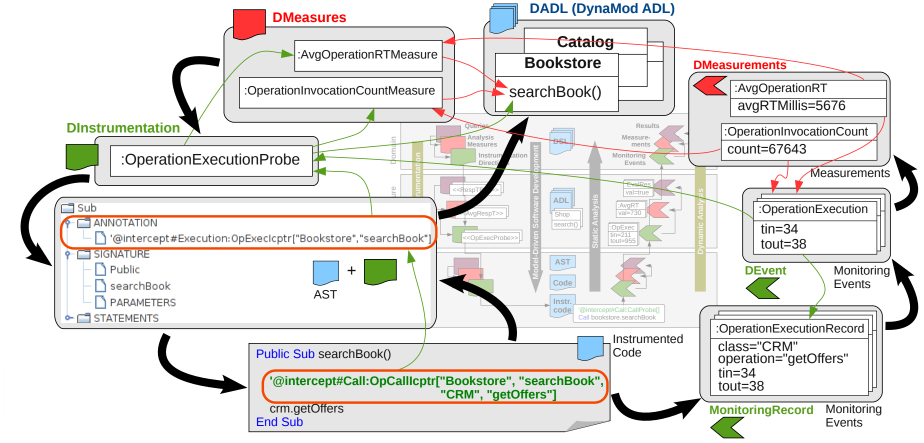
HSH NORDBANK
HSH Nordbank AG, Kiel
http://www.hsh-nordbank.de/

Model-Driven Instrumentation & Analysis

[van Hoorn et al. 2011b]

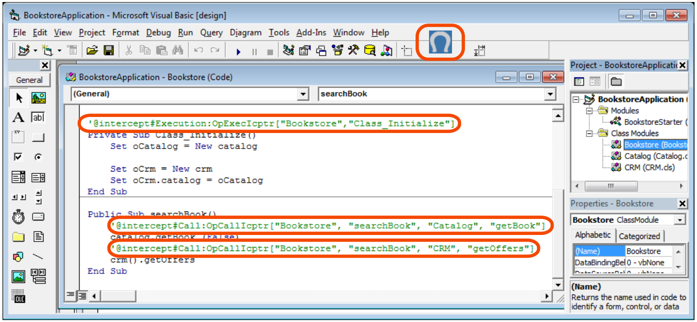


Model-Driven Instrumentation & Analysis (cont'd)



AOP-Based Instrumentation of VB6

[van Hoorn et al. 2011b]

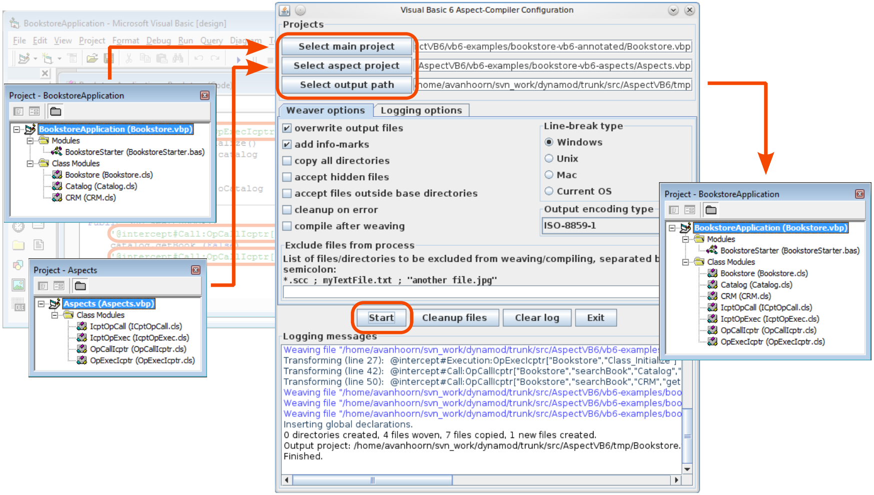

BookstoreApplication - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
General
BookstoreApplication - Bookstore (Code)
(General)
searchBook
Project - BookstoreApplicati
BookstoreApplication (
Modules
BookstoreStarter (
Class Modules
Bookstore (Bookst
Catalog (Catalog.c
CRM (CRM.cls)
Properties - Bookstore
Bookstore ClassModule
Alphabetic
Categorized
(Name) Bookstore
DataBindingBel 0 - vbNone

```
'@intercept#Execution:OpExecIcptr["Bookstore","Class_Initialize"]
Private Sub Class_Initialize()
    Set oCatalog = New catalog

    Set oCrm = New crm
    Set oCrm.catalog = oCatalog
End Sub

Public Sub searchBook()
    '@intercept#Call:OpCallIcptr["Bookstore", "searchBook", "Catalog", "getBook"]
    catalog.getBook (False)
    '@intercept#Call:OpCallIcptr["Bookstore", "searchBook", "CRM", "getOffers"]
    crm().getOffers
End Sub
```

(Name)
Returns the name used in code to identify a form, control, or data

AOP-Based Instrumentation of VB6

[van Hoorn et al. 2011b]


BookstoreApplication - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram
Project - BookstoreApplication
BookstoreApplication (Bookstore.vbp)
Modules
BookstoreStarter (BookstoreStarter.bas)
Class Modules
Bookstore (Bookstore.cls)
Catalog (Catalog.cls)
CRM (CRM.cls)
Project - Aspects
Aspects (Aspects.vbp)
Class Modules
IcptOpCall (ICptOpCall.cls)
IcptOpExec (IcptOpExec.cls)
OpCallIcptr (OpCallIcptr.cls)
OpExecIcptr (OpExecIcptr.cls)
'@intercept#Call:OpCallIcptr[
catalog.getBook (False)
'@intercept#Call:OpCallIcptr[
Project - BookstoreApplication
BookstoreApplication (Bookstore.vbp)
Modules
BookstoreStarter (BookstoreStarter.bas)
Class Modules
Bookstore (Bookstore.cls)
Catalog (Catalog.cls)
CRM (CRM.cls)
IcptOpCall (ICptOpCall.cls)
IcptOpExec (IcptOpExec.cls)
OpCallIcptr (OpCallIcptr.cls)
OpExecIcptr (OpExecIcptr.cls)
Start
Cleanup files
Clear log
Exit

Visual Basic 6 Aspect-Compiler Configuration

Projects

Select main project	ctVB6/vb6-examples/bookstore-vb6-annotated/Bookstore.vbp
Select aspect project	AspectVB6/vb6-examples/bookstore-vb6-aspects/Aspects.vbp
Select output path	home/avanhoorn/svn_work/dynamod/trunk/src/AspectVB6/tmp

Weaver options | Logging options

- [✓] overwrite output files
- [✓] add info-marks
- [ ] copy all directories
- [ ] accept hidden files
- [ ] accept files outside base directories
- [ ] cleanup on error
- [ ] compile after weaving

Line-break type: [✓ Windows] Unix Mac Current OS

Output encoding type: ISO-8859-1

Exclude files from process

List of files/directories to be excluded from weaving/compiling, separated b semicolon:

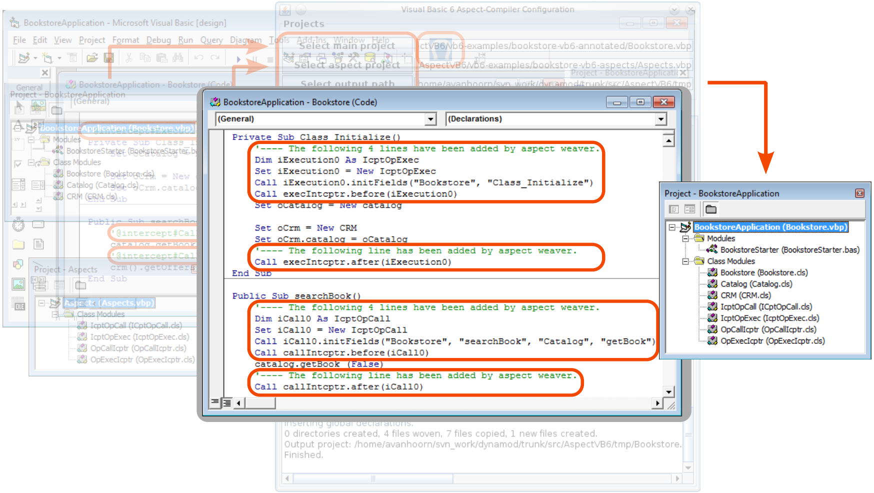
*.scc ; myTextFile.txt ; "another file.jpg"

Logging messages

```
Weaving file "/home/avanhoorn/svn_work/dynamod/trunk/src/AspectVB6/vb6-example
Transforming (line 27):  @intercept#Execution:OpExecIcptr["Bookstore","Class_Initialize"]
Transforming (line 42):  @intercept#Call:OpCallIcptr["Bookstore","searchBook","Catalog","
Transforming (line 50):  @intercept#Call:OpCallIcptr["Bookstore","searchBook","CRM","get
Weaving file "/home/avanhoorn/svn_work/dynamod/trunk/src/AspectVB6/vb6-examples/boo
Weaving file "/home/avanhoorn/svn_work/dynamod/trunk/src/AspectVB6/vb6-examples/boo
Weaving file "/home/avanhoorn/svn_work/dynamod/trunk/src/AspectVB6/vb6-examples/boo
Inserting global declarations.
0 directories created, 4 files woven, 7 files copied, 1 new files created.
Output project: /home/avanhoorn/svn_work/dynamod/trunk/src/AspectVB6/tmp/Bookstore.
Finished.
```

AOP-Based Instrumentation of VB6

[van Hoorn et al. 2011b]


BookstoreApplication - Microsoft Visual Basic [design]
File Edit View Project Format Debug Run Query Diagram Tools Add-Ins Window Help
Visual Basic 6 Aspect-Compiler Configuration
Projects
Select main project
Select aspect project
Select output path
Project - Aspects
Aspects (Aspects.vbp)
Class Modules
IcptOpCall (ICptOpCall.cls)
IcptOpExec (IcptOpExec.cls)
OpCallIcptr (OpCallIcptr.cls)
OpExecIcptr (OpExecIcptr.cls)

```
ctVB6/vb6-examples/bookstore-vb6-annotated/Bookstore.vbp
AspectVB6/vb6-examples/bookstore-vb6-aspects/Aspects.vbp
home/avanhoorn/svn_work/dynamod/trunk/src/AspectVB6/tmp
```

BookstoreApplication - Bookstore (Code)

(General) | (Declarations)

```
Private Sub Class_Initialize()
    '---- The following 4 lines have been added by aspect weaver.
    Dim iExecution0 As IcptOpExec
    Set iExecution0 = New IcptOpExec
    Call iExecution0.initFields("Bookstore", "Class_Initialize")
    Call execIntcptr.before(iExecution0)
    Set oCatalog = New catalog

    Set oCrm = New CRM
    Set oCrm.catalog = oCatalog
    '---- The following line has been added by aspect weaver.
    Call execIntcptr.after(iExecution0)
End Sub

Public Sub searchBook()
    '---- The following 4 lines have been added by aspect weaver.
    Dim iCall0 As IcptOpCall
    Set iCall0 = New IcptOpCall
    Call iCall0.initFields("Bookstore", "searchBook", "Catalog", "getBook")
    Call callIntcptr.before(iCall0)
    catalog.getBook (False)
    '---- The following line has been added by aspect weaver.
    Call callIntcptr.after(iCall0)
```

Project - BookstoreApplication

```
BookstoreApplication (Bookstore.vbp)
  Modules
    BookstoreStarter (BookstoreStarter.bas)
  Class Modules
    Bookstore (Bookstore.cls)
    Catalog (Catalog.cls)
    CRM (CRM.cls)
    IcptOpCall (ICptOpCall.cls)
    IcptOpExec (IcptOpExec.cls)
    OpCallIcptr (OpCallIcptr.cls)
    OpExecIcptr (OpExecIcptr.cls)
```

```
Inserting global declarations.
0 directories created, 4 files woven, 7 files copied, 1 new files created.
Output project: /home/avanhoorn/svn_work/dynamod/trunk/src/AspectVB6/tmp/Bookstore.
Finished.
```

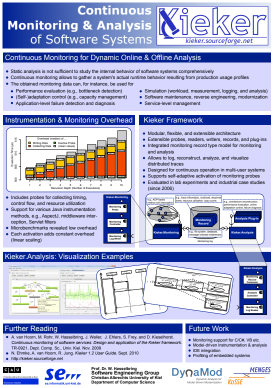
Further Reading & Current Activities

Further Reading

- 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. TR-0921, Dept. Comp. Sc., Univ. Kiel. Nov. 2009
- N. Ehmke, A. van Hoorn, R. Jung. Kieker 1.3 User Guide. May 2011
- `http://kieker.sourceforge.net`

Current and Future Activities

- Model-driven instrumentation & analysis
- Combination with static analysis
- Monitoring support for .NET, Visual Basic etc.
- Analysis IDE
- Additional architectural view types
- Integration with other analysis tools (e.g., Software-EKG, Fraunhofer's SAVE)
- Self-* & cloud applications
- **In the process of becoming a SPEC research tool**



Continuous Monitoring & Analysis of Software Systems

Kieker kieker.sourceforge.net

Continuous Monitoring for Dynamic Online & 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)
  - (Self-)adaptation control (e.g., capacity management)
  - Application-level failure detection and diagnosis
  - Simulation (workload, measurement, logging, and analysis)
  - Software maintenance, reverse engineering, modernization
  - Service-level management

Instrumentation & 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
- Microbenchmarks revealed low overhead
- Each activation adds constant overhead (linear scaling)

Kieker Framework

- Modular, flexible, and extensible architecture
- Extensible probes, readers, writers, records, and plug-ins
- 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
- Supports self-adaptive activation of monitoring probes
- Evaluated in lab experiments and industrial case studies (since 2006)

Kieker.Monitoring
Monitoring Probe
Monitoring Controller
Monitoring Log Writer
e.g., AOP-based method call interception
e.g., trace information, workload, response times, resource utilization, loop counts
e.g., architecture reconstruction, performance evaluation, online adaptation control, failure diagnosis
Monitoring Record
Analysis Plug-In
Kieker.Analysis
e.g., file system, database, message-oriented middleware
Monitoring log
Monitoring Record Consumer
Analysis Controller
Monitoring Log Reader

Kieker.Analysis: Visualization Examples

Further Reading

- 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.* TR-0921, Dept. Comp. Sc., Univ. Kiel. Nov. 2009
- N. Ehmke, A. van Hoorn, R. Jung. *Kieker 1.2 User Guide.* Sept. 2010
- http://kieker.sourceforge.net

Future Work

- Monitoring support for C/C#, VB etc.
- Model-driven instrumentation & analysis
- IDE integration
- Profiling of embedded systems

se.informatik.uni-kiel.de

Prof. Dr. W. Hasselbring
Software Engineering Group
Christian Albrechts University of Kiel
Department of Computer Science

DynaMod Dynamic Analysis for Model-Driven Modernization

MENGES KoSSE

Literature

codecentric GmbH. Performance survey 2008. http://www.codecentric.de/export/sites/www/_resources/pdf/performance-survey-2008-web.pdf, Mar. 2009.

B. Cornelissen, A. Zaidman, A. van Deursen, L. Moonen, and R. Koschke. A systematic survey of program comprehension through dynamic analysis. *IEEE Transactions on Software Engineering*, 35(5):684–702, 2009. ISSN 0098-5589.

S. Duszynski, J. Knodel, and M. Lindvall. SAVE: Software Architecture Visualization and Evaluation. In *Proceedings of the 2009 European Conference on Software Maintenance and Reengineering*, pages 323–324, Washington, DC, USA, 2009. IEEE Computer Society.

J. Ehlers and W. Hasselbring. Self-adaptive software performance monitoring. In R. Reussner, M. Grund, A. Oberweis, and W. F. Tichy, editors, *Software Engineering 2011: Fachtagung des GI-Fachbereichs Softwaretechnik*, volume 183 of *Lecture Notes in Informatics*, pages 51–62. GI, 2011.

N. Ehmke, A. van Hoorn, and R. Jung. Kieker 1.3 user guide. https://se.informatik.uni-kiel.de/kieker/documentation/, May 2011.

A. van Hoorn, W. Hasselbring, and M. Rohr. Engineering and continuously operating self-adaptive software systems: Required design decisions. In G. Engels, R. Reussner, C. Momm, and S. Sauer, editors, *Design for Future 2009: Proceedings of the 1st Workshop of the GI Working Group „Long-Living Software Systems (L2S2)"*, volume 537 of *CEUR Workshop Proceedings*, pages 52–63, Nov. 2009a. URL http://ceur-ws.org/Vol-537/.

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. 2009b. URL http://www.informatik.uni-kiel.de/uploads/tx_publication/vanhoorn_tr0921.pdf.

A. van Hoorn, S. Frey, W. Goerigk, W. Hasselbring, H. Knoche, S. Köster, H. Krause, M. Porembski, T. Stahl, M. Steinkamp, and N. Wittmüss. DynaMod project: Dynamic analysis for model-driven software modernization. In A. Fuhr, W. Hasselbring, V. Riediger, M. Bruntink, and K. Kontogiannis, editors, *Joint Proceedings of the 1st International Workshop on Model-Driven Software Migration (MDSM 2011) and the 5th International Workshop on Software Quality and Maintainability (SQM 2011)*, volume 708 of *CEUR Workshop Proceedings*, pages 12–13, Mar. 2011a. Invited paper.

A. van Hoorn, H. Knoche, W. Goerigk, and W. Hasselbring. Model-driven instrumentation for dynamic analysis of legacy software systems. In *Proceedings of the 13. Workshop Software-Reengineering (WSR '11)*, pages 26–27, 2011b. URL http://www.uni-koblenz-landau.de/koblenz/fb4/institute/uebergreifend/sre/conferences/wsr/wsr2011/wsr2011_proceedings.pdf.

J. Weigend, J. Siedersleben, and J. Adersberger. Dynamische Analyse mit dem Software-EKG. *Informatik-Spektrum*, pages 1–12, 2011.

C. Wulf. Runtime visualization of static and dynamic architectural views of a software system to identify performance problems. B.Sc. Thesis, CAU, AG Software Engineering, 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)*. IEEE, July 2011. To appear.