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)
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)
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

Dynamic Analysis & Continuous Monitoring

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

Services

Application

Middleware Container

Virtual Machine

Operating System

Hardware

Business/service monitoring

Application monitoring

Infrastructure monitoring

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, ...

Wilhelm Hasselbring, André van Hoorn (CAU)

Architecture Reconstruction via Dynamic Analysis

June 29, 2011 @ SEACON 2011
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
Java monitoring largely unknown.

[codecentric GmbH 2009]
Continuous Monitoring — Design Decisions

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; → AOP
Agenda

1 Dynamic Analysis & Continuous Monitoring

2 Kieker — Framework for Continuous Monitoring and Analysis

3 Architecture Reconstruction and Visualization Examples

4 Dynamic Analysis of Legacy Systems

5 Conclusions
**Monitoring & Analysis of Software Behavior**

**Kieker — Framework for Continuous Monitoring and Analysis**

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

```java
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**

**Instrumentation**

**Monitoring log**

**Statistical analysis**

Wilhelm Hasselbring, André van Hoorn (CAU)  
Architecture Reconstruction via Dynamic Analysis  
June 29, 2011 @ SEACON 2011
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

Visualization of results

Instrumentation

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 — Framework for Continuous Monitoring and Analysis

Framework Overview

Kieker

http://kieker.sourceforge.net

Monitoring Log/Stream

e.g., file system, database, message-oriented middleware

e.g., trace information, workload, response times, resource utilization, loop counts

e.g., architecture reconstruction, performance evaluation, online adaptation control, failure diagnosis

Wilhelm Hasselbring, André van Hoorn (CAU)

Architecture Reconstruction via Dynamic Analysis

June 29, 2011 @ SEACON 2011
Examples: Monitoring Record, Probe, Plugin

Kieker — Framework for Continuous Monitoring and Analysis ➤ Framework Overview

Custom Monitoring Record: MyRTMonitoringRecord

```java
@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

```java
 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

Wilhelm Hasselbring, André van Hoorn (CAU)  Architecture Reconstruction via Dynamic Analysis  June 29, 2011 @ SEACON 2011 12 / 28
Characteristics & Features

Kieker — Framework for Continuous Monitoring and Analysis

Framework Overview

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

Wilhelm Hasselbring, André van Hoorn (CAU)  Architecture Reconstruction via Dynamic Analysis  June 29, 2011 @ SEACON 2011 13 / 28
Kieker.TraceAnalysis Tool

Architecture Reconstruction and Visualization Examples

Wilhelm Hasselbring, André van Hoorn (CAU)  Architecture Reconstruction via Dynamic Analysis  June 29, 2011 @ SEACON 2011
**Sequence diagrams**

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

(a) Assembly-level view

(b) Deployment-level view
1. Sequence diagrams
2. **Dynamic call trees**
3. Hierarchical calling dependency graphs
4. System model

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

(b) **Aggregated** deployment-level call tree
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
1. Sequence diagrams
2. Dynamic call trees
3. Hierarchical calling dependency graphs
4. System model (here: HTML representation)
Calling Networks
Clustering based on Calling (Frequency) Relationships [Zheng et al. 2011]
Architecture Reconstruction and Visualization Examples

Community structure
Layered structure (hierarchy)

Xi’an Jiaotong University, Shaanxi [Zheng et al. 2011]
3D Visualization with DyVis

[Wulf 2010]

Architecture Reconstruction and Visualization Examples

DyVis contains an implementation that enables reading KDM conform files. Thus, with an appropriate parser, e.g. KADis, DyVis can represent the static architecture of any program independent of the programming language.

DyVis uses Kieker's interface for reading dynamic log data. Thus, it is also possible to integrate it into Kieker. Moreover, one can make and save a snapshot to an image file to easily insert the given architecture into a presentation or paper.

Figure 15 shows a snapshot of DyVis that illustrates the statics and dynamics of the sample application iBATIS JPetStore.

Interaction

The user is able to zoom in and out the city by scrolling the mouse wheel accordingly. By pressing one of the keys W,A,S,D one can move forward, to the left, backwards and to the right, respectively. It is also possible to rotate the view by dragging.
Anomaly Detection and Diagnosis

[ Ehlers and Hasselbring 2011 ]

Architecture Reconstruction and Visualization Examples
Agenda

1. Dynamic Analysis & Continuous Monitoring
2. Kieker — Framework for Continuous Monitoring and Analysis
3. Architecture Reconstruction and Visualization Examples
4. Dynamic Analysis of Legacy Systems
5. Conclusions
Context — DynaMod Project

Dynamic Analysis of Legacy Systems

DynaMod

Dynamic Analysis for Model-Driven Software Modernization

Definition of Transformations

Dynamic Analysis

Code Generation

Static Analysis

Model-Based Testing

Evaluation

Wilhelm Hasselbring, André van Hoorn (CAU)
Architecture Reconstruction via Dynamic Analysis
June 29, 2011 @ SEACON 2011
Context — DynaMod Project (cont’d)

Dynamic Analysis of Legacy Systems

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)

Read More:

- [http://kosse-sh.de/dynamod/](http://kosse-sh.de/dynamod/)
- MDSM @ CSMR 2011 [van Hoorn et al. 2011a]
Model-Driven Instrumentation & Analysis

[van Hoorn et al. 2011b]
Dynamic Analysis of Legacy Systems

Queries
Analysis Measures
Instrumentation Directives

DSL

Results
Measurements
Monitoring Events

<<RespTReq>>

<<AvgRespT>>

<<OpExecProbe>>

ADL

Static Analysis

Model-Driven Software Development

:EvalRes val=true

:AvgRT val=730

:OpExec tin=211 tout=955

AST Code DSL ADL Shop search()

Instr. code

'@intercept#Call:CallProbe[]

Call bookstore.searchBook

Wilhelm Hasselbring, André van Hoorn (CAU)

Architecture Reconstruction via Dynamic Analysis

June 29, 2011 @ SEACON 2011
Dynamic Analysis of Legacy Systems

Public Sub searchBook()
...
    crm.getOffers
End Sub

Code

e[]
k

AST
Catalog
Bookstore
searchBook()

DMeasures
:AvgOperationRTMeasure
:OperationInvocationCountMeasure

DInstrumentation
:OperationExecutionProbe

DADL (DynaMod ADL)

Catalog
Bookstore
searchBook()

DMeasures
:AvgOperationRT
avgRTMillis=5676

:OperationInvocationCount
count=67643

DMeasurements
t
:AvgOperationRT
avgRTMillis=5676

DEvent
Monitoring Events

:OperationExecutionRecord
tin=34
tout=38

:OperationExecution
tin=34
tout=38

DEvent
Monitoring Events

:OperationExecutionRecord
class="CRM"
operation="getOffers"
tin=34
tout=38

Wilhelm Hasselbring, André van Hoorn (CAU)
Architecture Reconstruction via Dynamic Analysis
June 29, 2011 @ SEACON 2011 26 / 28
AOP-Based Instrumentation of VB6

[van Hoorn et al. 2011b]

Dynamic Analysis of Legacy Systems

Architectural Reconstruction via Dynamic Analysis

June 29, 2011 @ SEACON 2011
AOP-Based Instrumentation of VB6
[van Hoorn et al. 2011b]
Dynamic Analysis of Legacy Systems

Wilhelm Hasselbring, André van Hoorn (CAU)
Architecture Reconstruction via Dynamic Analysis
June 29, 2011 @ SEACON 2011 27 / 28
Further Reading & Current Activities

Conclusions

Further Reading


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
Conclusions


