Model-driven Instrumentation to forecast Dynamic Applications
Kieker Palladio Days 2013

Reiner Jung    Robert Heinrich    Eric Schmieders

28.11.2013
Context iObserve Project

- Cloud-based SOA applications
- Multivariate optimization
- Forecasting of application performance
- Evaluation of data geo location policies
Common Component Modeling Example (CoCoME) [RRMP08]
Motivation

Introduction

**Code-based**

- Determining of instrumentation points in code by hand
- Model reconstruction for analysis
  - Missing information compared to design model
  - Algorithm requires construction knowledge

**Model-based**

- Specify observation of Architectural/Abscet level
- Single point a specification/technology independent
- Provides relationship of monitoring data to architectural elements
Solution Overview

Solution

IAL Artefact \rightarrow \text{operation references} \rightarrow \text{Palladio Comp. Model} \rightarrow \text{derivied from}

Design Time

TradingSystem. Application.Calc \rightarrow \text{generate instrumentation} \rightarrow \text{generate/map to code}

Run Time

TradingSystem. Application.DB

PCM Runtime Model \rightarrow \text{model updates} \rightarrow \text{kieker Analysis}

Analysis Model

Jung, Heinrich, Schmieders

Model-driven Instrumentation to forecast Dynamic Appl

28.11.2013 5 / 16
Solution Overview

Jung, Heinrich, Schmieders
Model-driven Instrumentation to forecast Dynamic Appl
28.11.2013
Solution Overview

Solution Design Time

- IAL Artefact
- Palladio Comp. Model
  - operation references
  - generate instrumentation
  - map to code

Solution Run Time

- TradingSystem. Application.Calc
- TradingSystem. Application.DB
- Database
- PCM Runtime Model
  - model updates
  - monitoring data

Analysis Model

- Analysis

Derivied from TradingSystem. Application.DB Analysis Model

Jung, Heinrich, Schmieders
Model-driven Instrumentation to forecast Dynamic Appl
28.11.2013 5 / 16
Solution Overview

Solution

Design Time

IAL Artefact

Palladio Comp. Model

derived from

Run Time

TradingSystem. Application.Calc

TradingSystem. Application.DB

Database

PCM Runtime Model

Analysis Model

monitoring data

model updates

Jung, Heinrich, Schmieders

Model-driven Instrumentation to forecast Dynamic Appl

28.11.2013 5 / 16
Solution Overview

Solution Design Time Run Time

TradingSystem. Application.Calc
TradingSystem. Application.DB

IAL Artefact
Palladio Comp. Model

Operation references
generate instrumentation
generate/map to code

model updates
monitoring data

PCM Runtime Model

Analysis

Analysis Model
dervied from

Jung, Heinrich, Schmieders
Model-driven Instrumentation to forecast Dynamic Appl
Solution Overview

Design Time

IAL Artefact → Operation references → Palladio Comp. Model

Generate instrumentation → Generate/map to code

Run Time

TradingSystem. Application.Calc → Database

TradingSystem. Application.DB

Derived from TradingSystem. Application.DB Analysis Model

Analysis Model

PCM Runtime Model

Model updates

Monitoring data

Kieker

Analysis

Jung, Heinrich, Schmieders
Model-driven Instrumentation to forecast Dynamic Appl

28.11.2013
**Solution Overview**

**Design Time**
- IAL Artefact
- operation references
- generate instrumentation
- Palladio Comp. Model
- generate/map to code
- derived from
- Trading System Application DB
- Trading System Application Calc

**Run Time**
- Database
- PCM Runtime Model
- Monitoring data
- Kieker Analysis

**Analysis Model**
Solution Overview

Solution

Design Time

IAL Artefact

operation references

generate instrumentation

Palladio Comp. Model

generate/map to code

derived from

Analysis Model

Run Time

TradingSystem. Application.Calc

TradingSystem. Application.DB

Database

PCM Runtime Model

model updates

monitoring data

Analysis

Jung, Heinrich, Schmieders  Model-driven Instrumentation to forecast Dynamic Appl
Design-Time Models

Solution

IAL Artefact
weaving and configuration
generate probes

Kieker.Monitoring

<<component>>
Monitoring Probe

<<component>>
Monitoring Controller

<<component>>
Monitoring Writer

Monitoring Log/Stream

generate record types

IRL Artefact

<<artifact>>
Monitoring Record

Jung, Heinrich, Schmieders
Model-driven Instrumentation to forecast Dynamic Appl
28.11.2013 6 / 16
Design-Time Models

Data structures

IAL Artefact
weaving and configuration

generate probes

<<component>>
Kieker.Monitoring

<<component>>
Monitoring Probe

<<component>>
Monitoring Controller

<<component>>
Monitoring Writer

<<artifact>>
Monitoring Record

Monitoring Log/Stream

IRL Artefact
generate record types

Jung, Heinrich, Schmieders
Model-driven Instrumentation to forecast Dynamic Appl
28.11.2013
Join-point model for monitoring

IAL Artefact

weaving and configuration

generate probes

<<component>>
Kieker.Monitoring

<<component>>
Monitoring Probe

<<component>>
Monitoring Controller

<<component>>
Monitoring Writer

Monitoring Log/Stream

generate record types

IRL Artefact

<<artifact>>
Monitoring Record

Jung, Heinrich, Schmieders
Model-driven Instrumentation to forecast Dynamic Appl
28.11.2013 6 / 16
Design-Time Models

Solution

IAL Artefact
weaving and configuration
generate probes

Kieker.Monitoring

Monitoring Probe

<<component>>

Monitoring Controller

Monitoring Writer

Monitoring Log/Stream

IAL Artefact

generate record types

IRL Artefact

Monitoring Log/Stream

IAL Artefact
Runtime Activities

Solution

- Data collection
  - Kieker monitoring [vHWH12]
- Data evaluation
  - Interval based evaluation
  - Utilizing MAMBA language [FvHJ+11]
- Updating Palladio runtime model [BKR07]
Analysis Chain

Solution

IAL Artefact

PCM Runtime Model

<<Processor>>
Map Generator

Model to Code

<<Filter>>
Rewrite

Analysis Model

PCM Runtime Model n

PCM Runtime Model n+1

Kieker Log

<<Filter>>
Select

<<Filter>>
Analysis

<<Filter>>
Model Update
Instrumentation Aspect Language

Languages

@communication SOAP

aspect TradingSystem.** * insert (**) {
  before EntryCallRecord (time,$signature,1)
  after ResponseRecord (time,$signature)
}

Select application model elements
@communication SOAP

aspect TradingSystem.** * insert (**) {
  before EntryCallRecord (time,$signature,1)
  after ResponseRecord (time,$signature)
}

Weaving Instruction, Record Type, Data Collection
@communication SOAP

aspect TradingSystem.** * insert (**) {
  before EntryCallRecord (time,$signature,1)
  after ResponseRecord (time,$signature)
}

Probe technology
Determining Program Code Join-Points

Source Model

IAL
Aspect
<<Component>>
Database
<<Interface>>
Access
<<Method>>
insert
query

Target Model/Code

Aspect Code

```
execution(tr.app.example.Database
void insert(Object obj))
```

Application Code

```
package tr.app.example
class Database implements Access {
    public void insert(Object obj) {
        ...
    }  

    public void query(Query q) {
        ...
    }
}
```
**Features**

- Primitive types: int, long, double, string, ...  
- Type system  
  - Record structures  
  - Multiple inheritance based on record fragments  
- Code generation infrastructure  
  - Record structure  
  - Serialization support (Kieker API)  
- Code generators for multiple languages
struct EntryCallRecord extends ParentRecord : MyPattern {
    const key NO_SIGNATURE = "<no-signature>"

    long time = -1
    key signature = NO_SIGNATURE
}

pattern MyPattern : Other1, Other2 {
    long example = 0
}
struct EntryCallRecord extends ParentRecord : MyPattern {
    const key NO_SIGNATURE = "<no-signature>"

    long time = -1
    key signature = NO_SIGNATURE
}

pattern MyPattern : Other1, Other2 {
    long example = 0
}
struct EntryCallRecord extends ParentRecord : MyPattern {
    const key NO_SIGNATURE = "<no signature>"

    long time = -1
    key signature = NO_SIGNATURE
}

pattern MyPattern : Other1, Other2 {
    long example = 0
}
struct EntryCallRecord extends ParentRecord : MyPattern {
    const key NO_SIGNATURE = "<no-signature>"

    long time = -1
    key signature = NO_SIGNATURE
}

pattern MyPattern : Other1, Other2 {
    long example = 0
}
```
struct EntryCallRecord extends ParentRecord : MyPattern {
    const key NO_SIGNATURE = "<no-signature>"

    long time = -1
    key signature = NO_SIGNATURE
}

pattern MyPattern : Other1, Other2 {
    long example = 0
}
```
IRL Current State

Languages

Implementation

- Editor, grammar and type-system (prototype)
- Code generation for Java, C and Perl

Open Issues

- Multiple definition of properties forbidden
- Convenience property renaming
  classSignature → calleeClassSignature
- Special operations, like refersToSameOperationAs
Summary

Conclusions

Presentation

- Approach covering design time and runtime modeling
- Modeling instrumentation and monitoring data structures
- Instrumentation: model to code mapping
- Mapping monitoring data back to an runtime model

Participation

- Project: http://build.se.informatik.uni-kiel.de/gitlab/iobserve/model-driven-instrumentation
- Blog articles: http://www.oiloftrop.de/tag/kieker/
- Contact: reiner.jung@email.uni-kiel.de
Future Work

Conclusions

Tooling and Language Tasks

1. Finalizing of the IRL typing system
2. Fixing build problem of IAL
3. IAL model reference to code mapping
4. Monitoring data to PCM runtime model mapping
5. Updating PCM usage model

Publication

1. Provide Eclipse update site
2. Documentation and tutorials
3. Integration in Kieker process

