SW-Archäologie mit AOP
(Praxisbericht)

Ausgrabungen vergangener SW-Architekturen

9. Workshop SW-Reengineering
Bad Honnef

Mail 2007
Oliver.Boehm@agentes.de

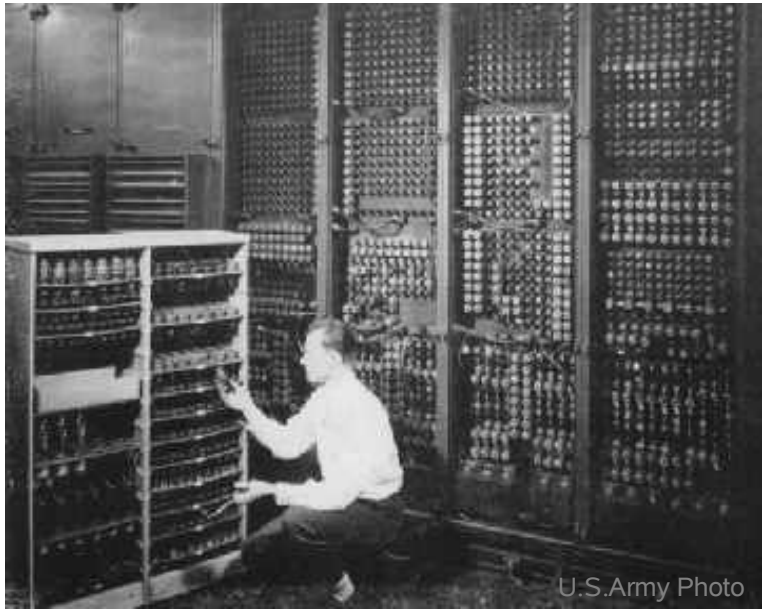
agentes AG 2006

- **Aktionäre**
  - Pironet NDH AG (Hauptaktionär)
  - Vorstände & Manager
- **Standorte**
  - Stuttgart (Firmensitz), Kassel, Hamburg, München
- **140 Mitarbeiter**
- **Ergebnis**
  - 2004: Umsatz 8,6 Mio EUR, EBIT 834 TEUR
  - 2005: Umsatz 8,7 Mio EUR, EBIT ca. 850 TEUR
  - 2006: Umsatz 9,7 Mio EUR (HGB)
- **Kernkompetenzen**
  - Softwareentwicklung für / Wartung von komplexen Verfahren und Banksystemen
  - Softwarelösungen zur Vertriebsoptimierung

Inhalt

- die klassische Herangehensweise
- Ein kleiner Ausflug mit AOP
- Herangehensweise mit AOP
- Fazit
- Diskussion

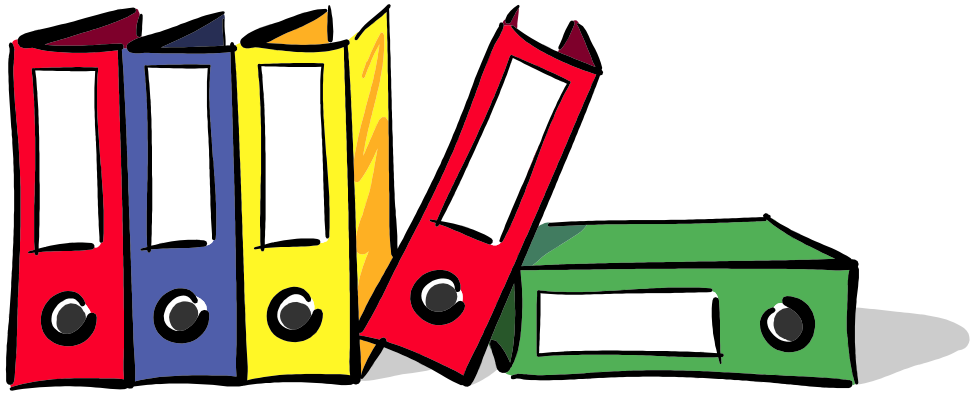
Die klassische Herangehensweise



Wunsch und Realität

- die Anwendung ist dokumentiert
- die Doku ist aktuell
- Test-System vorhanden
- Unit-Tests u. Integrationstests
- Testergebnisse reproduzierbar
- Know-How vorhanden

- Doku? Welche Doku?
- Doku wurde nie aktualisiert
- Test-System = Produktivsystem
- Test-System vorhanden, aber anderes Verhalten
- ehem. Entwickler verschwunden



Einstieg Dokumentation

- **Pflichtenheft**
- **Architektur-Dokumente**
  - hilfreich: Übersicht über die Architektur
  - Klassendiagramme und andere UML-Diagramme
- **Infrastruktur**
  - beteiligte Systeme
  - Firewalls

- **Probleme / Fragen**
  - Aktualität?
  - welche Dokumente gibt es?
  - wo liegen sie?
  - wer ist Ansprechpartner?

- **Die aktuell gültigige Quelle ist oft der Source-Code!**

Alles was mir hilft, den Sourcecode zu verstehen, hilft mir das System zu verstehen:
- **IDE (Navigation)**
- **Reverse Engineering (statische Sicht)**

Test-Systeme

- **Testfälle als Ergänzung zur Doku vorhanden?**
  - Unit-Tests
  - Integrationstests
- **Test-System**
  - ganz wichtig für das Verständnis
  - aber: nicht immer im Zugrff
  - oder: nicht immer vorhanden
- **Problem Qualität:**
  - Sind Testfälle noch aktuell?
  - Wie aussagekräftig?
  - Werden die wichtigen Fälle abgedeckt?

Teställe dokumentiereren die dynamische Sicht des Systems.

Ein kleiner Abstecher in AOP

Beispiel: Bankkonto

fachliche Concerns

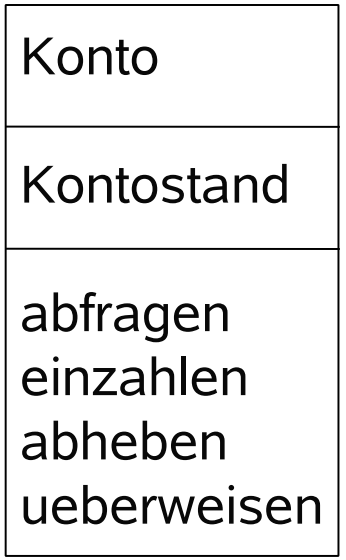
- Einzahlung
- Auszahlung
- Überweisung
- Bonitätsprüfung
- ...

nichtfachliche Concerns

- Logging
- Performance
- Authorisierung
- Sicherheit
- ...

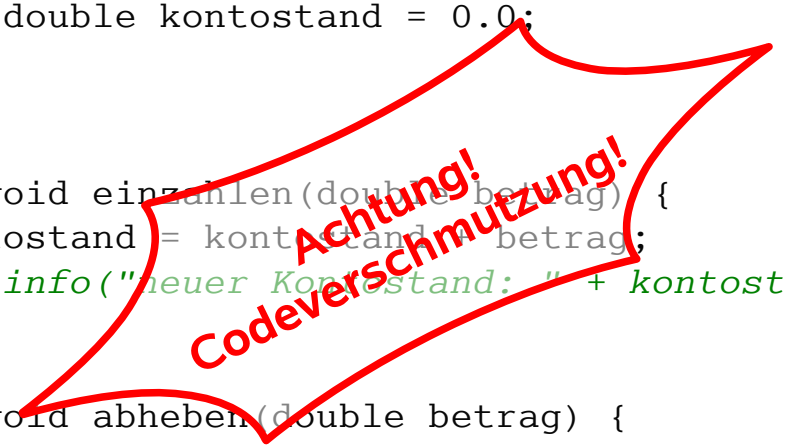
Ein Fallbeispiel...

```
public class Konto {

    private double kontostand = 0.0;

    public double abfragen() {
        return kontostand;
    }

    public void einzahlen(double betrag) {
        kontostand = kontostand + betrag;
    }

    public void abheben(double betrag) {
        kontostand = kontostand - betrag;
    }

    public void ueberweisen(double betrag, Konto anderesKonto) {
        this.abheben(betrag);
        anderesKonto.einzahlen(betrag);
    }

}
```



...mit Logging

```
public class Konto {

    private static Logger log = Logger.getLogger(Konto.class)
    private double kontostand = 0.0;

    ...

    public void einzahlen(double betrag) {
        kontostand = kontostand + betrag;
        log.info("neuer Kontostand: " + kontostand);
    }

    public void abheben(double betrag) {
        kontostand = kontostand - betrag;
        log.info("neuer Kontostand: " + kontostand);
    }

    ...

}
```



neue Anforderung:

alle Konto-Bewegungen müssen protokolliert werden!

...mit AspectJ

```
public aspect LogAspect {

    private static Logger log = Logger.getLogger(LogAspect.class);

    after(double neu) : set(double Konto.kontostand) && args(neu) {
        log.info("neuer Kontostand: " + neu);
    }

}
```

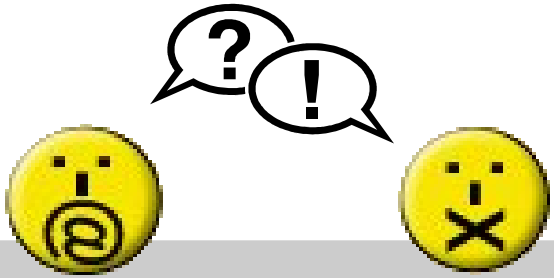
Konto
Kontostand
abfragen einzahlen abheben ueberweisen

LogAspect

Do you speak AspectJ?

agentes

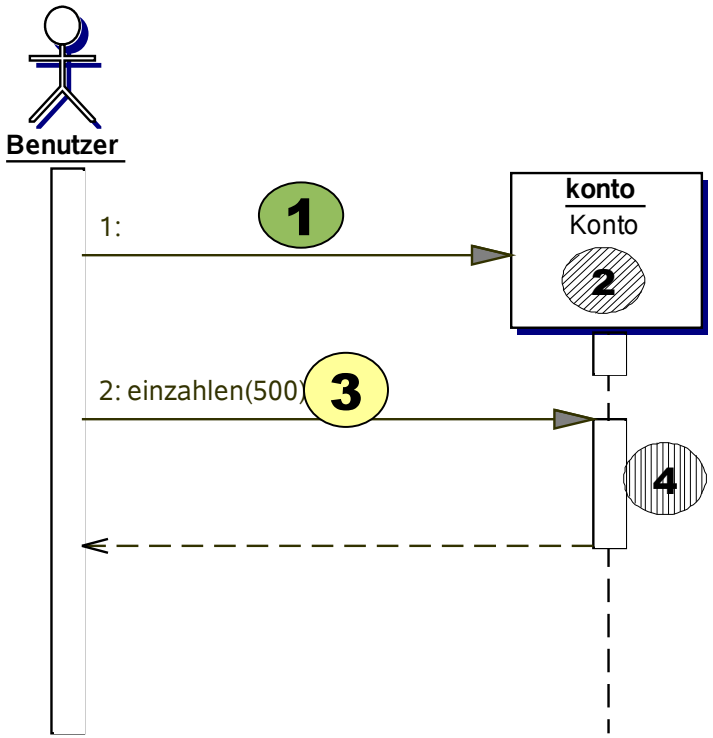
- **Joinpoint**
  - Eingriffspunkte im Programm, an denen Code erweitert oder modifiziert werden soll
- **Pointcut**
  - eine Auswahl von Joinpoints
- **Advice**
  - der Code für den Pointcut
- **Introduction**
  - Erweiterung anderer Klassen, Interfaces, Aspekte um zusätzliche Funktionalität
- **Aspect**
  - Konstrukt, in dem das obige abgelegt wird



Joinpoints können sein

- Aufrufen einer Methode
- Ausführen einer Methode
- Zugriff auf eine Variable
- Behandeln einer Exception
- Initialisierung einer Klasse
- Initialisierung eines Objekts

```
Konto konto = new Konto();
konto.einzahlen(500.0);
```



(1) Konstruktor aufrufen

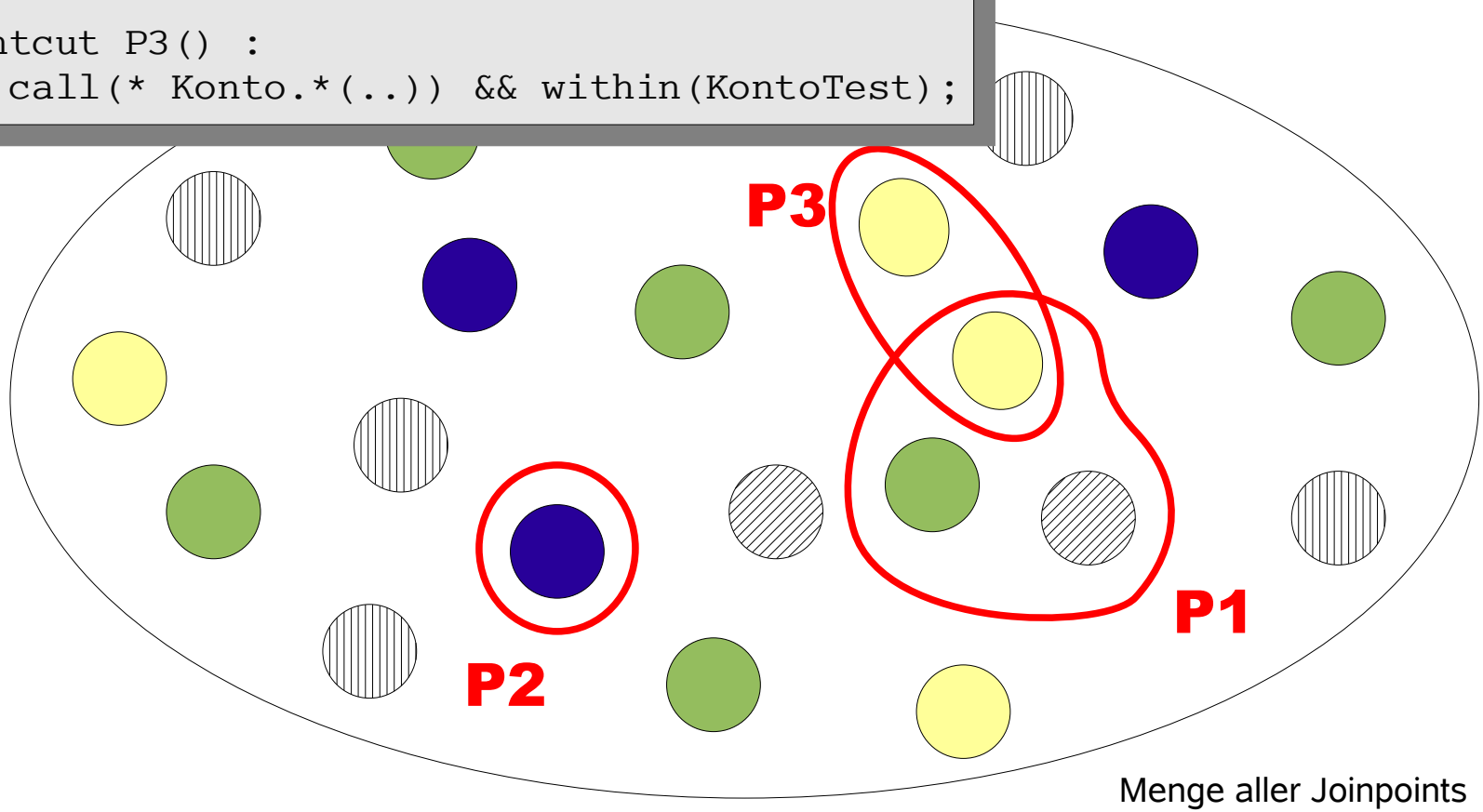
(2) Objekt initialisieren

(3) Methode aufrufen

(4) Methode ausführen

Pointcuts

```
pointcut P2() : set (double Konto.kontostand)

pointcut P3() :
    call(* Konto.*(..)) && within(KontoTest);
```



Menge aller Joinpoints

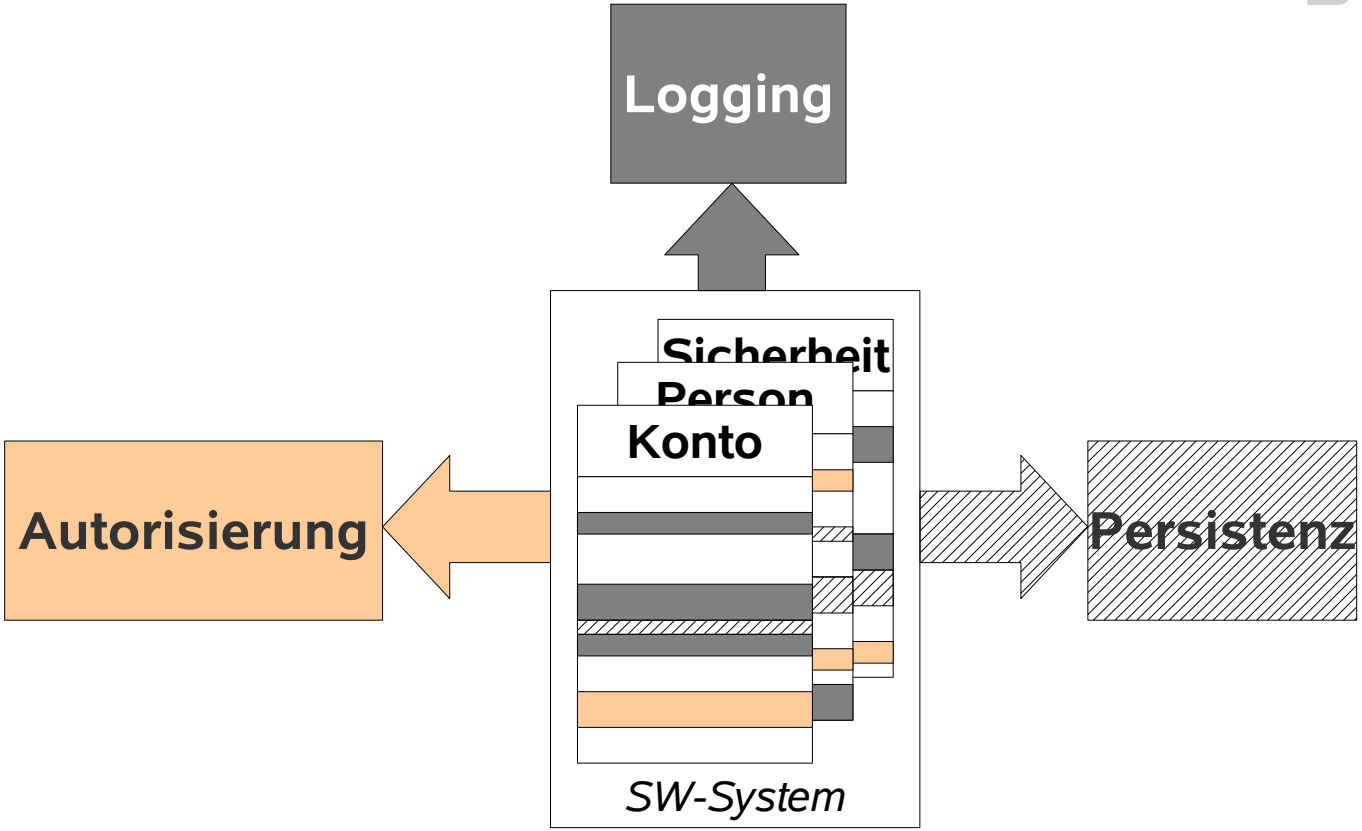
Advice

- **Code, der eingewebt wird**
  - before()
  - after()
  - around()
- **Ähnlichkeit mit Methoden**

```
after() : P3() {
    log.info("TEST: " + thisJoinPoint);
}
```

```
TEST: call(void bank.Konto.einzahlen(double))
TEST: call(double bank.Konto.abfragen())
...
```

Do you think in AOP?



Das System als Menge von “Concerns”

Der Webe-Vorgang (Weaving)

Konto

```
private double kontostand = 0.0;

public void einzahlen(double betrag) {
    kontostand = kontostand + betrag;
}
public void abheben(double betrag) {
    kontostand = kontostand - betrag;
}
public void ueberweisen(double betrag, Konto anderesKonto) {
    this.auszahlen(betrag);
    anderesKonto.einzahlen(betrag);
}
```

KontoAspect

```
pointcut kontobewegung(double betrag):
    (call(public void bank.Konto.einzahlen(double))
        || call(public void bank.Konto.abheben(double)))
    && args(betrag);

before(double betrag) : kontobewegung(betrag) {
    if (betrag < 0) {
        throw new IllegalArgumentException("negativer Betrag");
    }
}
```

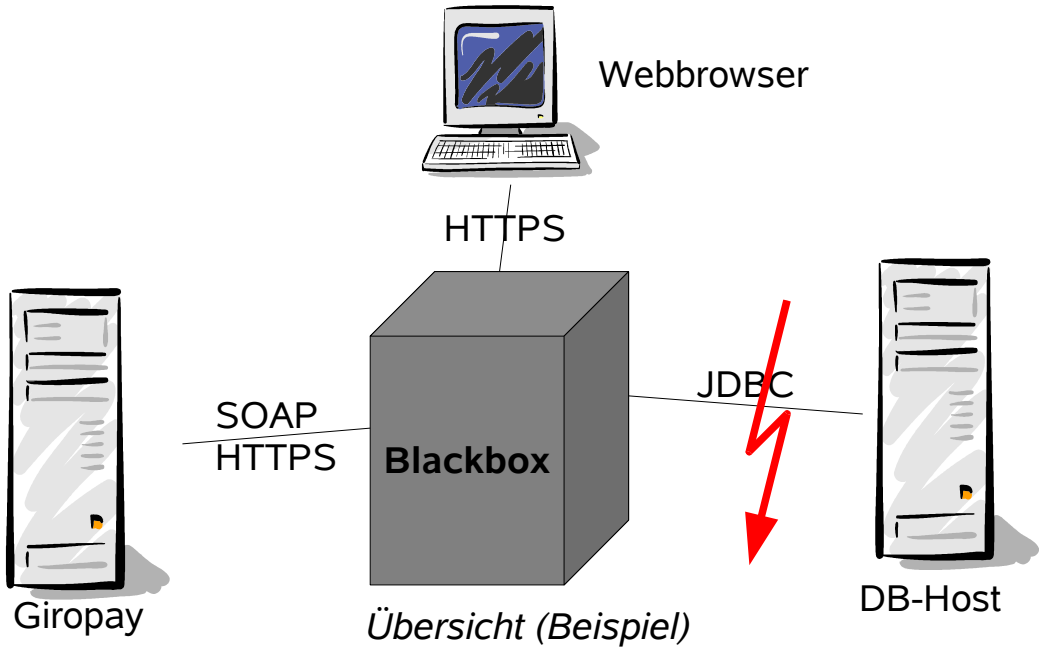
Konto.class

```
private double kontostand = 0.0;
public void einzahlen(doube betrag) {
    if (betrag < 0) {
        throw new IllegalArgumentException("negativer Betrag");
    }
    kontostand = kontostand + betrag;
}
public void abheben(double betrag) {
    if (betrag < 0) {
        throw new IllegalArgumentException("negativer Betrag");
    }
    kontostand = kontostand - betrag;
}
public void ueberweisen(double betrag, Konto anderesKonto) {
    this.auszahlen(betrag);
    anderesKonto.einzahlen(betrag);
}
```

agentes

Herangehensweise mit AOP

Schnittstellen beobachten



Übersicht (Beispiel)

```
java.net.SocketException: java.net.ConnectException: Connection refused
    at com.mysql.jdbc.StandardSocketFactory.connect(StandardSocketFactory.java:156)
    at com.mysql.jdbc.MysqlIO.<init>(MysqlIO.java:283)
    at com.mysql.jdbc.Connection.createNewIO(Connection.java:2541)
    at com.mysql.jdbc.Connection.<init>(Connection.java:1474)
    at com.mysql.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:264)
    at java.sql.DriverManager.getConnection(DriverManager.java:525)
    at java.sql.DriverManager.getConnection(DriverManager.java:193)
    at bank.Archiv.init(Archiv.java:25)
    ...
```

DB-Verbindungen

```
after() returning(Object ret) :
          call(* java.sql.*.*(..)) {
    log.debug(getAsString(thisJoinPoint) + " = " + ret);
}
```

```
java.sql.DriverManager.getConnection("jdbc:mysql://localhost:3306/test") = com.mysql.jdbc.Connection(
java.sql.Connection.prepareStatement("SELECT kontostand from konto WHERE kontonr = ?;") =
            com.mysql.jdbc.ServerPreparedStatement[1] - SELECT kontostand from konto WHERE kon
java.sql.PreparedStatement.setInt(1, 1) = (null)
java.sql.PreparedStatement.executeQuery() = com.mysql.jdbc.ResultSet@183f74d
java.sql.ResultSet.next() = true
java.sql.ResultSet.getDouble(1) = 5055.94
java.sql.ResultSet.close() = (null)
```

Analog können auch andere Verbindungen / Aufrufe überwacht werden!

Aufruf von außen (1)

```
public pointcut executePublic() :
    (execution(public * bank..*.*(..))
        || execution(public bank..*.new(..)))
    && !within(EnvironmentAspect);

public pointcut executeFramework() :
    execution(* bank..*.*(..)) || execution(bank..*.new(..));

public pointcut calledFromOutside() :
    executePublic() && !cflowbelow(executeFramework());

before() : calledFromOutside() {
    Signature sig = thisJoinPoint.getSignature();
    String caller =
        getCaller(Thread.currentThread().getStackTrace(), sig);
    log.info(caller + " calls " + sig);
}
```

alle Aufrufe, die public sind, aber nicht aus dem Framework kommen

Aufruf von außen (2)

org.apache.jsp.index_jsp._jspService(index_jsp.java:54) calls bank.Konto(int)
org.apache.jsp.index_jsp._jspService(index_jsp.java:58) calls void bank.Konto.einzahlen(double)
org.apache.jsp.index_jsp._jspService(index_jsp.java:60) calls double bank.Konto.abfragen()
org.apache.jsp.index_jsp._jspService(index_jsp.java:54) calls bank.Konto(int)
org.apache.jsp.index_jsp._jspService(index_jsp.java:58) calls void bank.Konto.einzahlen(double)
org.apache.jsp.index_jsp._jspService(index_jsp.java:60) calls double bank.Konto.abfragen()

Daten-Recorder

- **Idee:**
  - statt Logging-Ausgabe Objekte in „Objekt-Log“ abspeichern
- **mehrereRealisierungsmöglichkeiten:**
  - ObjectOutputStream
  - XMLSerializer
  - oder eigene Lösung
- **später:**
  - Objekte wieder einspielen

```
after() returning(Object ret) : sqlCall() {
    objLogger.log(thisJoinPoint);
    objLogger.log(ret);
}
```

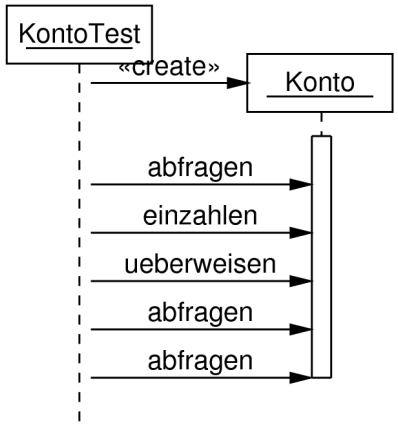
Aufnahme einspielen

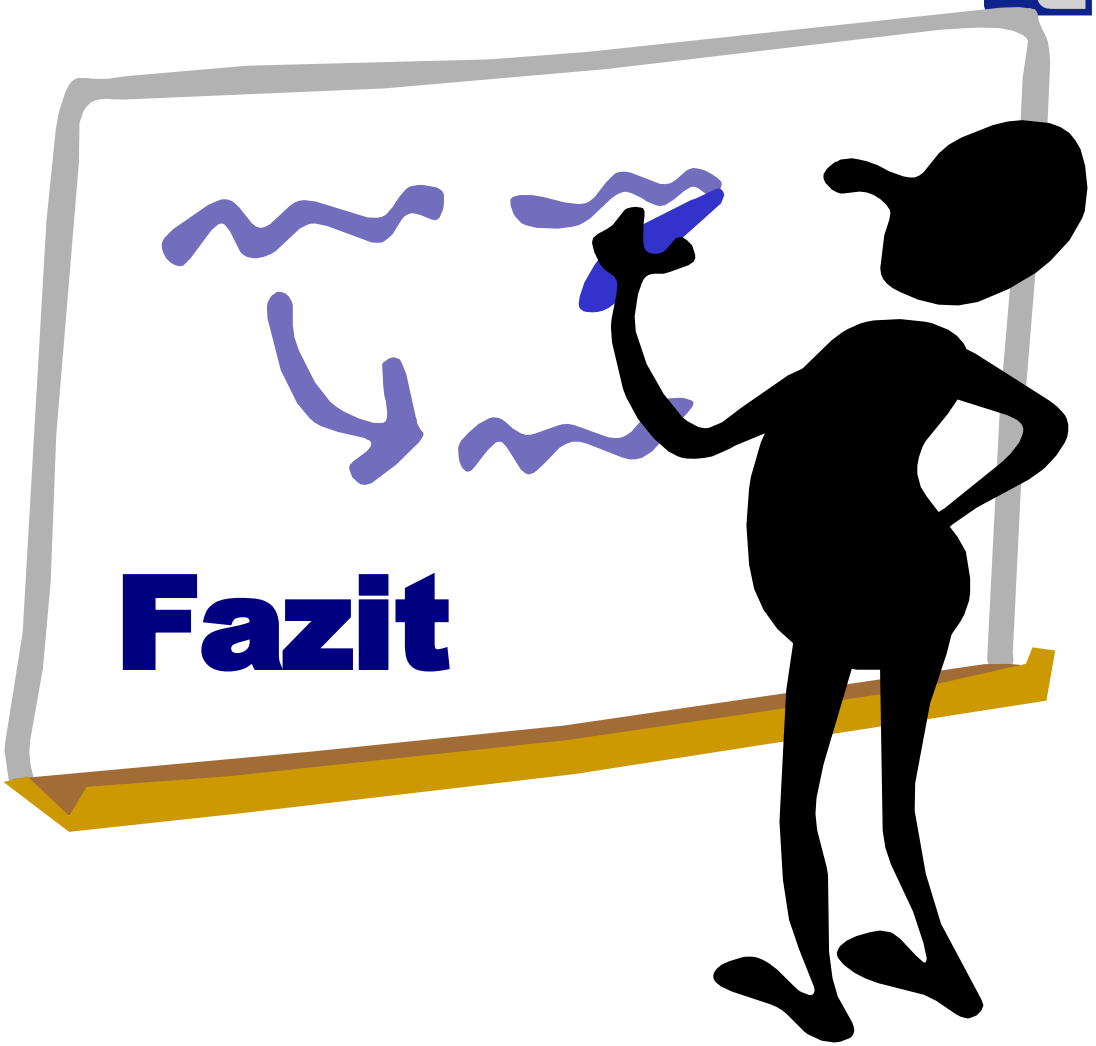
- **(einfache) Lösung:**
  - Hashtable aufbauen
    - key: JoinPoint,
    - value: abgespeicherte Return-Values
  - einspielen:

```
Object around() : sqlCall() {
    Object logged = logAnalyzer.getReturnValue(thisJoinPoint);
    return logged;
}
```

Ergebnis

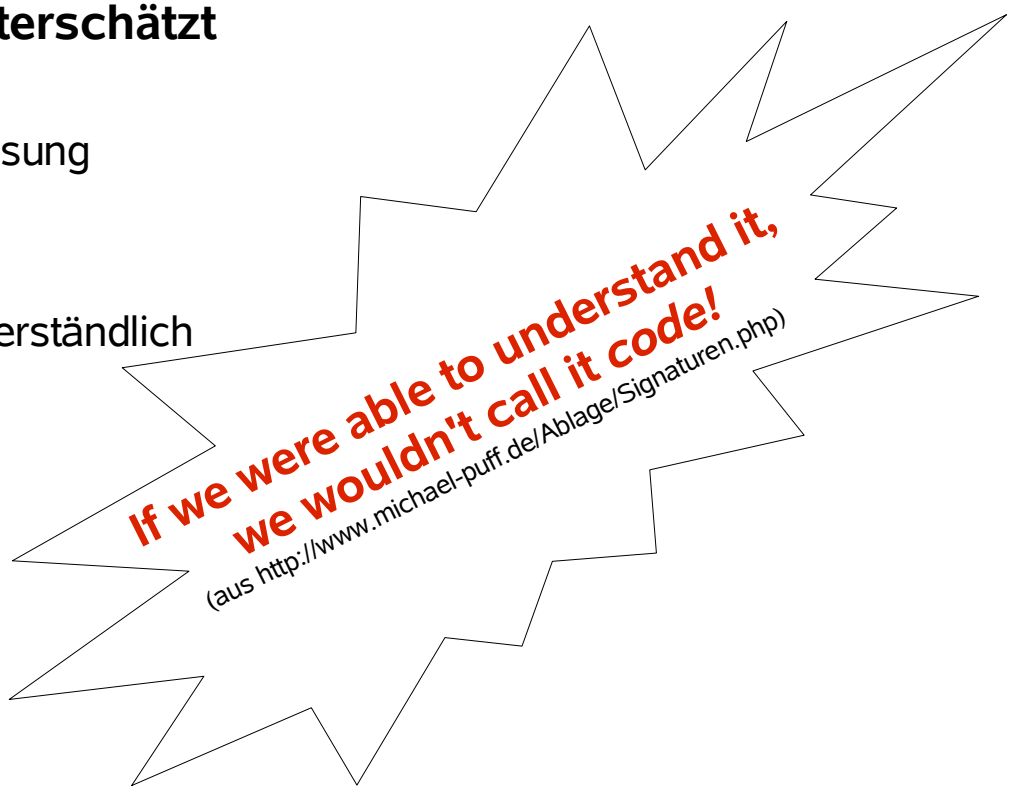
- **Tests laufen jetzt auch ohne Datenbank**
- **evtl. Anpassungen nötig**
- **Code-Änderungen:**
  - können in Aspekte ausgelagert werden
  - mehrere Möglichkeiten der Einbindung
    - statisch (zur Compile-Zeit)
    - dynamisch (Loadtime-Weaving)
- **mögl. Erweiterungen:**
  - Sequenz-Diagramme generieren (z.B. mit UMLGraph + GNU plotutils)
  - HealthCheck





Alt-Anwendungen

- **Aufwand für Einarbeitung oft unterschätzt**
- **Dokumentation**
  - Teil des Problems und nicht der Lösung
- **Source**
  - einzig aktuelle Dokumentation
  - aber: oft ausgewuchert und nicht verständlich
- **Testfälle sind oft Problemfälle**
  - oftmals scheitert es an Testsystem



AOP kann helfen...

- zusätzliche Log-Möglichkeiten einzubauen,
- Schnittstellen zu überwachen,
- Objekt-Recorder zu implementieren und implantieren,
- die aufgenommenen Objekte wieder einzuspielen,
- Änderungen und Erweiterungen vom Code zu trennen,
- u.v.m.

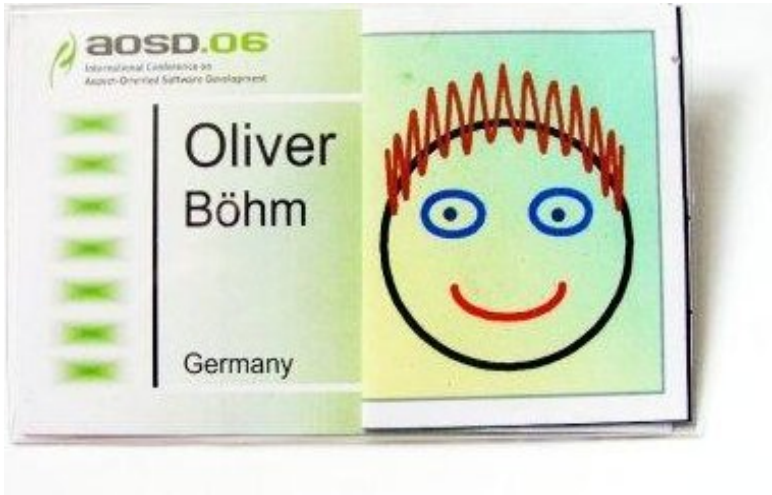
Aber:

- AOP ist nur ein Werkzeug (kein Allheilmittel)

Links & Literatur

- Oliver Böhm
  Aspektorientierte Programmierung mit AspectJ 5
  http://www.dpunkt.de/buch/3-89864-330-1.html
- Media-Streamer in Java
  ein Fallbeispiel mit AOP
  http://www.agentes.de/index.php?id=122#c258
- UMLGraph
  http://www.spinellis.gr/sw/umlgraph/

Vielen Dank



agentes AG

Oliver Böhm

oliver.boehm@agentes.de

Telefon 0711/2012-2734
Telefax 0711/2012-6734

Räpplenstaße 17
70191 Stuttgart