

# Aspekte im Projektalltag

## Erfahrungen mit AspectJ

Ludger Solbach

19.04.2007

# Agenda

## Projektvorstellung

- Ziele, Setup, Architektur, Tools
- Beispiel Rechenkern

## Einsatz von AspectJ im Projekt

- Entwicklung und QS, Infrastruktur, Fachlichkeit
- Beispiel Rechenkern/Fixing Aspect

## Erfahrungen

- Entwicklung, Build, Prozess, Team

## Fazit

# Projektvorstellung Branchenprognose

## Projektziele

- Erstellung volkswirtschaftlicher Prognosen für Sparkassenmitarbeiter
  - branchenspezifisch, regionalisiert
- Einbindung der Berichte in bestehende Web-Anwendung
- Berechnung der Prognosen mittels makro-ökonomischem Modell
  - Ca. 50.000 Zeilen C++, halbjährliche Updates
- Redaktionssystem für die textuelle Kommentierung der Prognose durch Branchenexperten (1. Release Ende April)

## Projektsetup

- Gesamtprojekt ca. 25 Personen an 3 Standorten
- Entwicklungsteam 6 Personen im DSV
- Vorgehen nach Rational Unified Process
  - Prototyping, Highest Risk First
  - Anforderungsänderungen spät im Projekt

# Projektvorstellung

## Architektur, Design

- Komponentenarchitektur, Multilayer
- Fachliches Design (Domain Driven Design)
  - Entities, Factories, Repositories, Services im Domain Layer, Controller im Application Layer
  - Generische Subdomains (z.B. Generischer Rechenkern, Content Management)

## Technologien und Tools

- Java 5, AspectJ 1.5, Eclipse 3.2, AJDT 1.4
- Spring 2.0, Tomcat 5.5
- Ant, CruiseControl
  - JUnit, Fit/Fitnessse
  - Cobertura, JDepend, PMD/CPD, Findbugs, Checkstyle

# Beispiel Generischer Rechenkern

## Allgemeine Anforderungen

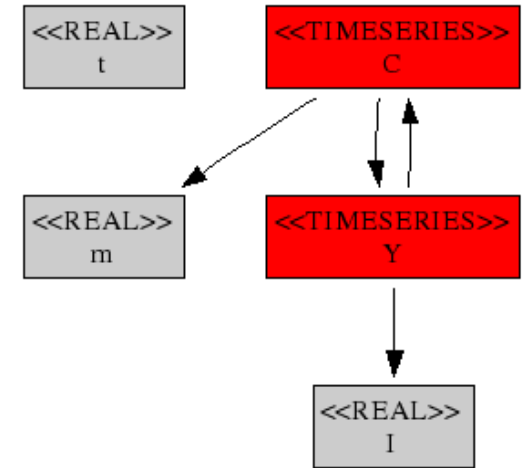
- Mathematische Datentypen
  - Reelle Zahlen, Vektoren, Matrizen, Zeitreihen
- Mathematische Funktionen
- Modelle als Daten
- Verschiedene Berechnungsarten
  - Hierarchisch, Sequentiell, Iterativ, Zeitreihen
  - Einfache Konvergenzkriterien

## Spezifische Anforderungen für die Prognoseberechnung

- Korrektur von Schätzabweichungen bei der Prognose von Zeitreihen (RhoAdjust)
- Übersteuern von Berechnungsergebnissen zur Szenarioberechnung (Fixing)
- Komplexe Konvergenzkriterien, abhängig von der Anzahl der Iterationen

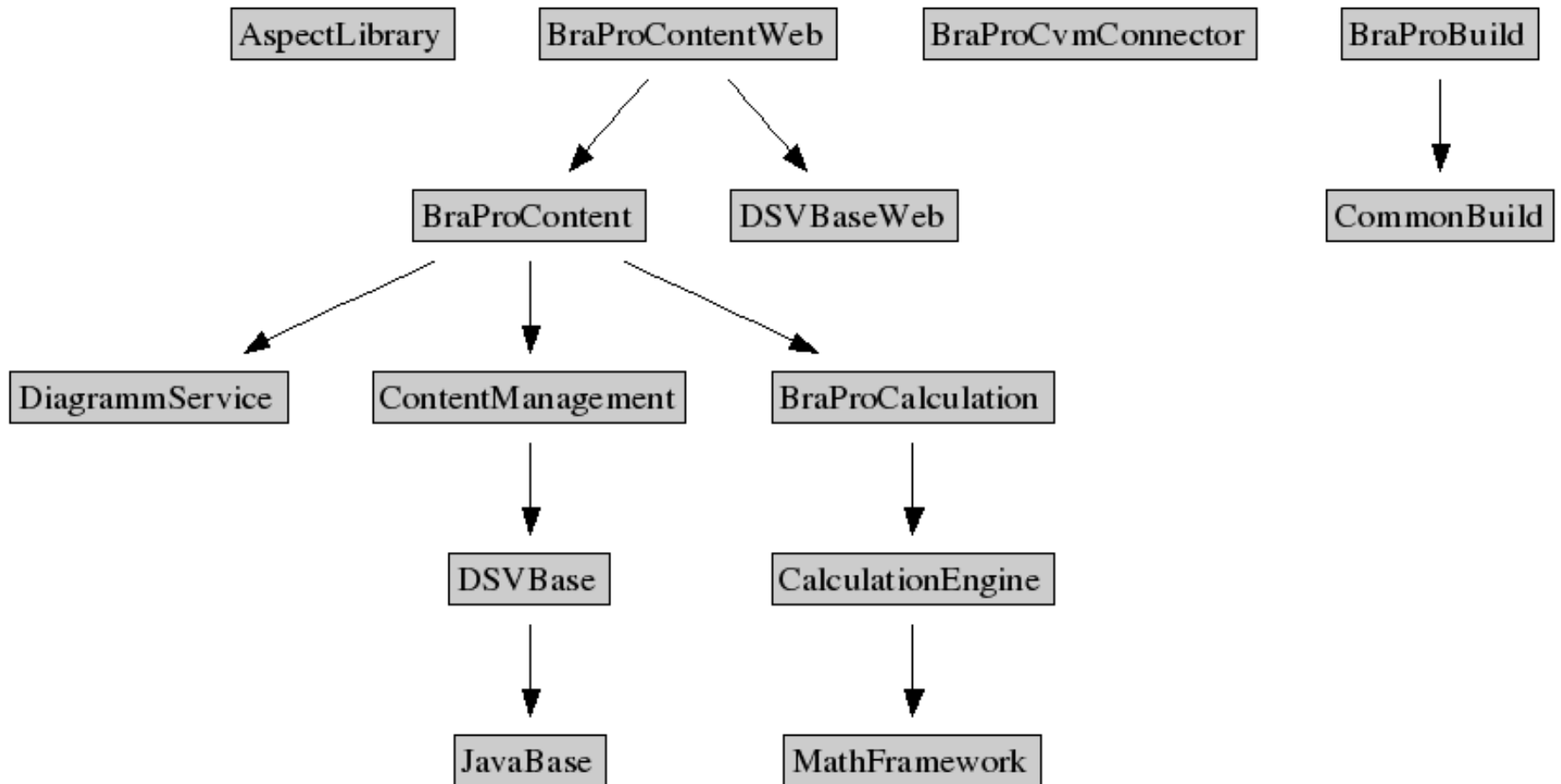
# Beispiel Generischer Rechenkern

```
<calculation-model name="Konsum" type="ITERATIVE_TIMESERIES"
  maxit="100" start="0" steps="5">
  <var name="t" type="REAL" dir="IN" desc="Zeitindex"/>
  <var name="m" type="REAL" dir="IN" desc="Multiplikator"/>
  <var name="I" type="REAL" dir="IN" desc="Investition"/>
  <var name="C" type="TSERIES" dir="INOUT" desc="Konsum"/>
  <var name="Y" type="TSERIES" dir="INOUT" desc="Einkommen"/>
  <eqn name="C">
    <invar name="m"/>
    <invar name="Y"/>
    <text> C[t] = 100 + m * Y[t-1] </text>
  </eqn>
  <eqn name="C">
    <invar name="I"/>
    <invar name="C"/>
    <text> Y[t] = I + C[t] </text>
  </eqn>
</calculation-model>
```



# Projektstruktur

## Grafik CVS-Module



# Einsatz von AspectJ im Projekt

## Einsatz von AspectJ im Projekt

- **Entwicklungsaspekte**
  - Vereinfachung von Implementierung und Debugging
- **QS-Aspekte**
  - Validierung von Architektur und Design
- **Infrastrukturaspekte**
  - Kapselung technischer Querschnittsfunktionalität
- **Fachliche Aspekte**
  - Funktionale Erweiterungen oder Änderungen generischer Komponenten aufgrund spezieller Anforderungen

# Beispiele für Aspekte

## Beispiele Entwicklung/QS

- Tracing
- Profiling
- Erzeugen von Sequenzdiagrammen
- Testunterstützung (Mock-Objects, Fehlererzeugung)
- Architektur-Validierung
- Validierung von Design-Richtlinien

# Beispiele für Aspekte

## Beispiele für Infrastrukturaspekte

- Logging
- Caching
  - Rechenergebnisse
  - Repositories
- Exception Conversion
  - System Exceptions in Domain Exceptions
- Entities
  - Introductions von Default-Konstruktoren, equals(), hashCode()
- DB-Transactionshandling

# Beispiele für fachliche Aspekte

## Spezifische Anpassungen bestehender Funktionalitäten

- **IndexAccessAspect**
  - Anpassung des Zugriffs auf mehrdimensionale Daten (0-/1-basiert)
- **ConvergenceAspect**
  - Behandlung von komplexen Konvergenzkriterien
- **ExceptionAspekt**
  - Behandlung von undefinierten Berechnungsergebnissen (Weiterrechnen oder Exception bei NaN oder Infinity)

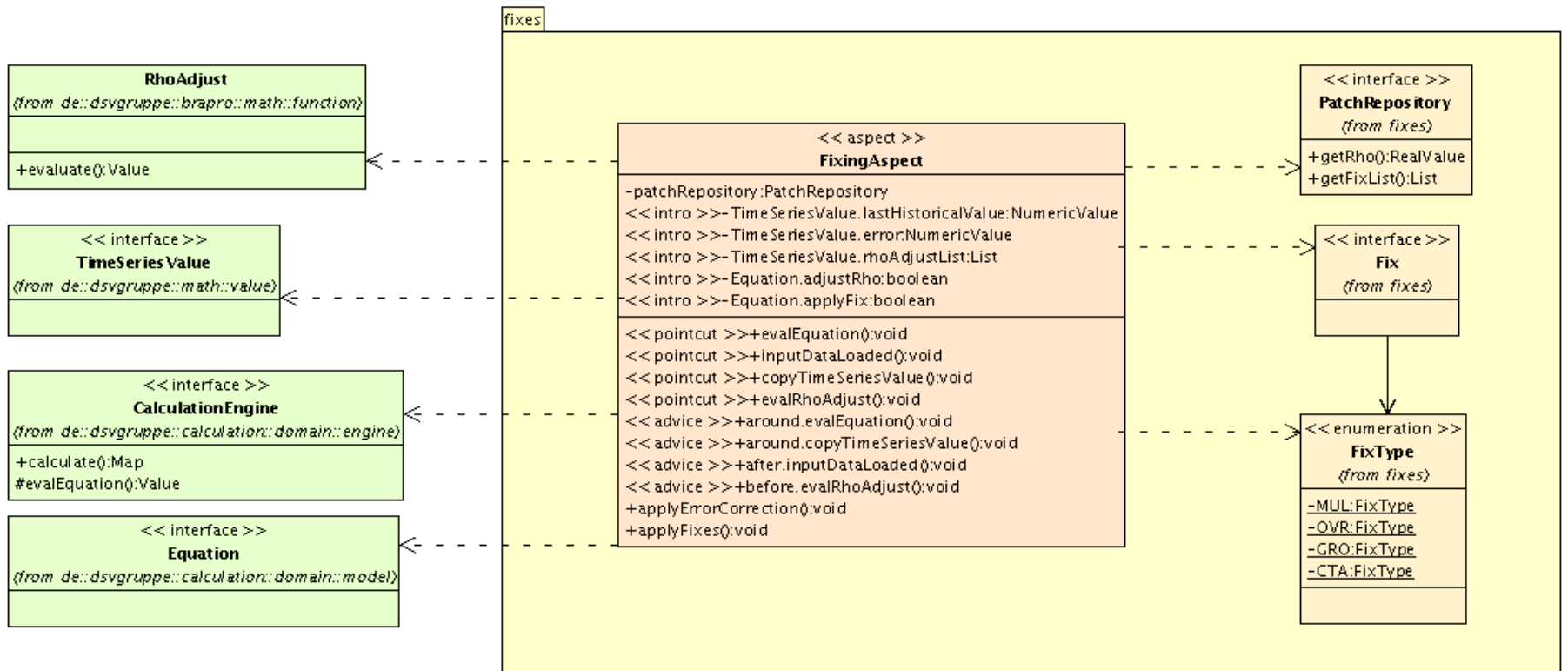
## Erweiterungen der Funktionalität

- **SpecialMathAspect**
  - Spezielle (math. undefinierte) Operationen für makroökon. Berechnungen
- **FixingAspect**
  - Korrekturen bei der Prognose von Zeitreihen, Handling von Szenarioberechnungen

# Beispiel Rechenkern/FixingAspect

## Anatomie des FixingAspect

cd: fixes.1



# Erfahrungen in der Entwicklung

## Probleme

- Semantik der Pointcuts nicht einfach
  - call vs. execution, static vs. dynamic typing
- Ausgaben von QS-Tools, die auf Bytecode arbeiten
  - Cycles in JDepend, wenn Aspekte nicht im gleichen Package liegen
- Probleme mit XML-Serialisierung per XStream
  - Kryptische Namen von Introduced Fields, Aliasdefinitionen nötig
  - Introduced Fields sind transient -> Sonderbehandlung
- Eclipse
  - Exceptions wegen OutOfPermSpace -> maxPermSpace auf 128MB
  - Java-Editor zeigt z.T. Phantomfehler an (Introductions, AspectJ-Editor auch für .java)
  - Debugging wird komplexer

# Erfahrungen beim Build-Prozess

## Integration von AspectJ in den Ant Build

- Erfordert strukturierten, modularisierten Buildprozess
- Switch für die Verwendung des AspectJ oder Java Compilers
- Initialaufwand einplanen
- Komplexe Pfade für den iajc-Task (Classpath, Aspectpath, Inpath, Sourcepath)
  - nicht trivial
  - Alle Klassen aus Jars im Inpath landen im Output, nicht nur durch Aspekte betroffene Klassen

# Erfahrungen beim Entwicklungsprozess

## Entwicklungsprozess

- Es gibt bis jetzt wenig Erfahrungen bzgl. der Integration in SD-Prozesse
- Requirements Engineering
  - Identifizierung von Aspekten
  - Modellierung der Use Cases (Extension Points, Extensions)
- Analyse & Design
  - Modellierung der Aspekte nicht standardisiert (UML-Profil)
  - Tools können mit Aspekten nicht umgehen
- Test
  - Aspekte können die Tests vereinfachen, z.B. Fehlersituationen erzeugen

# Erfahrungen beim Entwicklerteam

## Entwicklerteam

- Im Team muss Akzeptanz für AOP geschaffen werden
  - Nutzen aufzeigen und Begeisterung wecken durch konkrete Beispiele
- Es müssen nicht alle Entwickler im Team Aspekte programmieren können
  - Alle Teammitglieder sollten Konzepte/Grundwissen kennen
  - Die Entwicklung von Aspekten setzt gute und interessierte Entwickler voraus (Abstraktionsfähigkeit, Syntax, Lernkurve)
  - Normale Entwickler brauchen sich (fast) nicht um Querschnittsaspekte kümmern

## Fazit

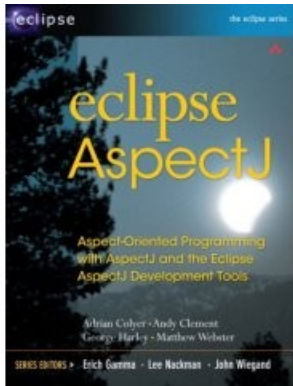
- AspectJ ist ein Werkzeug
  - weder "silver bullet" noch "golden hammer"
  - Mittel zum Zweck, Einsatz dort, wo sinnvoll

## Erreichte Ziele durch den AspectJ Einsatz

- Deutliche Vereinfachung der Implementierung
  - Verbesserte Modularisierung
- Wiederverwendbare Komponenten (generische Subdomains)
  - Variationen und Erweiterungen lassen sich in Aspekten abbilden
  - Kapselung von spezieller Funktionalität

## Lernbereiche

- Sprachmöglichkeiten, Pattern und Idiome
- Integration von AOSD in den frühen Projektphasen

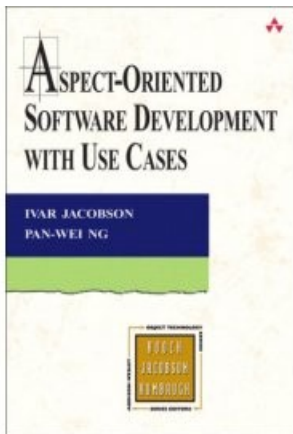


## **Eclipse AspectJ**

Aspect-Oriented Programming with AspectJ and the Eclipse AJDT

Adrian Colyer, Andy Clement, George Hartley, Matthew Webster

Addison-Wesley, ISBN 0-321-24587-3



## **Aspect-Oriented Software Development with Use Cases**

Ivar Jacobson, Pan-Wei Ng

Addison-Wesley, ISBN 0-321-26888-1

***Der Einsatz von AspectJ ist nicht umsonst,  
aber es lohnt sich!***



Deutscher  
Sparkassenverlag

## Ludger Solbach

Berater für Softwareentwicklung  
Softwareentwicklung  
Geschäftssparte Systemhaus

Deutscher Sparkassen  
Verlag GmbH  
Am Wallgraben 115  
70565 Stuttgart

Telefon +49 711 782-1840  
Telefax +49 711 782-1690  
[ludger.solbach@dsv-gruppe.de](mailto:ludger.solbach@dsv-gruppe.de)