canoo

› your provider for business web solutions ›

Sanierung grosser Software Systeme

Sibylle Peter, Dr. Dieter Holz
Canoo Engineering AG
Basel, Switzerland

Sibylle Peter

Dieter Holz

Inhalt

- Einführung
- Die Fallstudie: Investionsberatung für Kundenberater
- Die Sanierung
- Wie wir arbeiten
- Wie wird ein Sanierungsprojekt ein Erfolg?

Einführung

Sanierungsprojekte

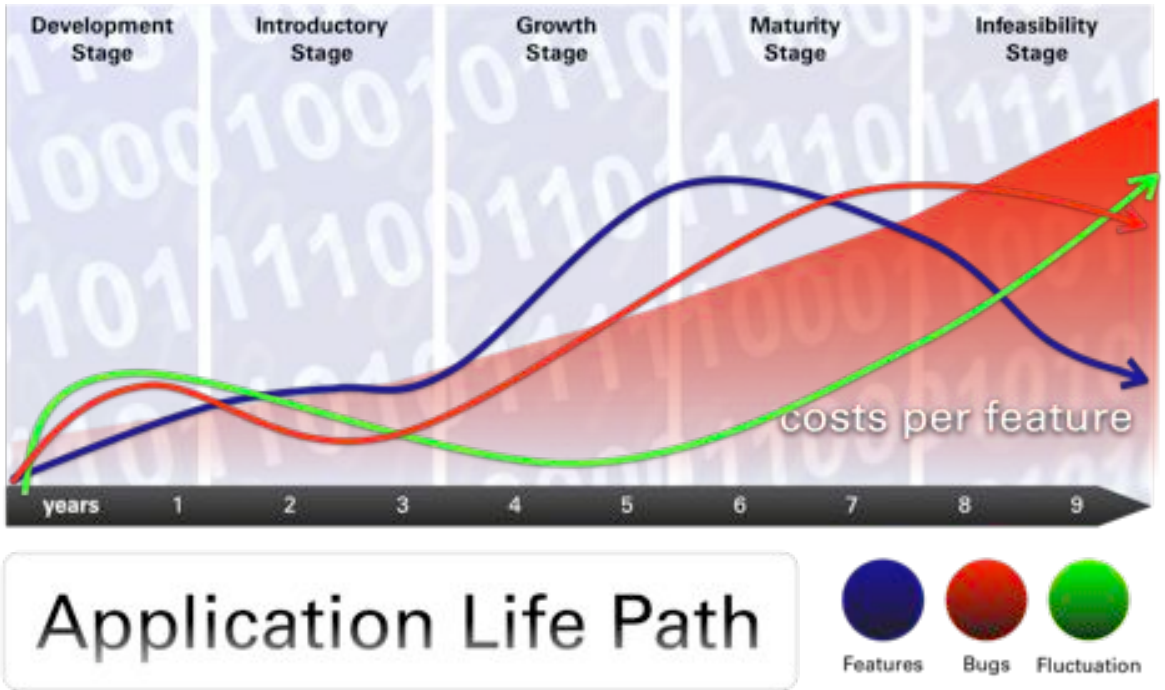
Sanierungsprojekte

Sanierung durch Refactoring

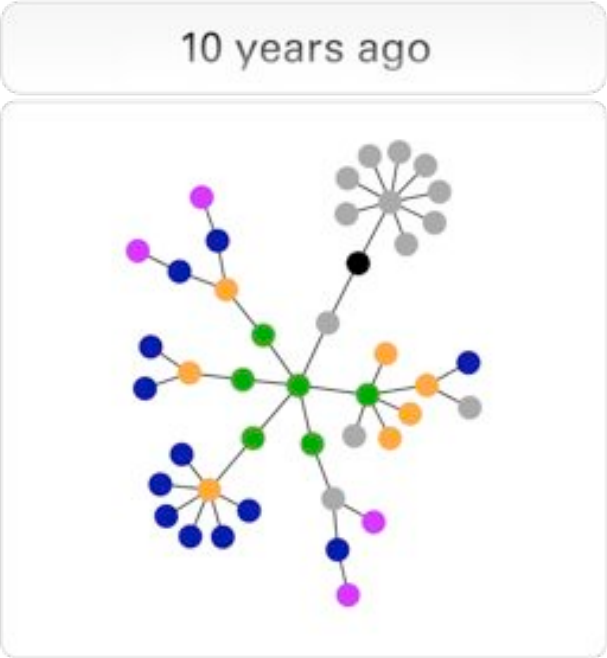
Definition (Martin Fowler, Refactoring, pg. 53):

Refactoring (noun): *a change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior.*

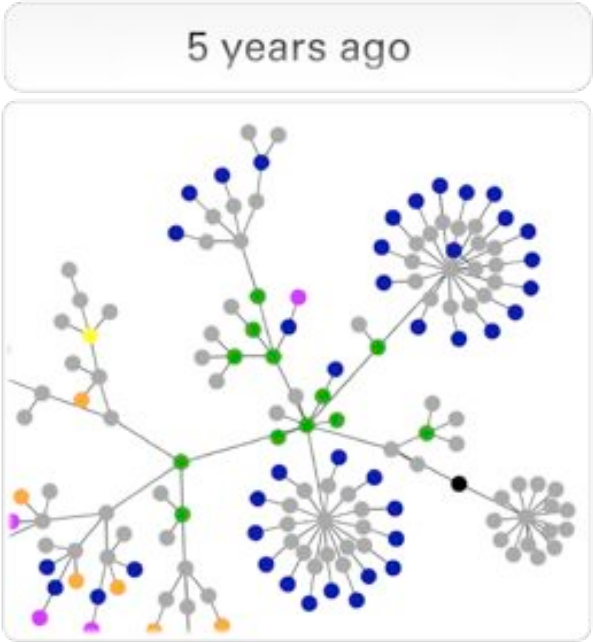
Refactor *(verb)*: *To restructure software by applying a series of refactorings without changing it's observable behavior*


Development Stage
Introductory Stage
Growth Stage
Maturity Stage
Infeasibility Stage
costs per feature
years
1
2
3
4
5
6
7
8
9
Application Life Path
Features
Bugs
Fluctuation

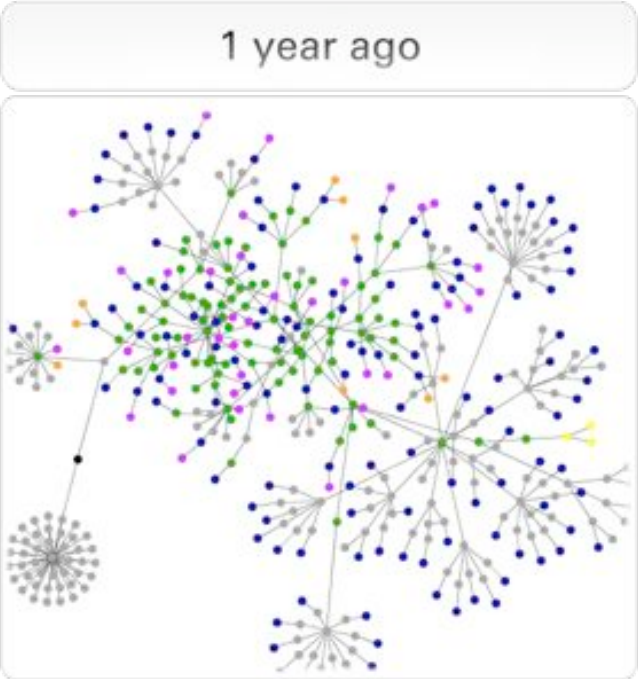
Software Entropie



Software Entropie



Software Entropie



Die Fallstudie

TIS Settings | PNC | Client/Portfolio | In /Outflow | Asset Allocation | Cash&Trx | Structure Analysis | Show Positions | Load Trx | Load Model

Asset Allocation — Investment Summary

Investment Strategy	Income	
Reference Currency	CHF	
Value	⦿ Percentage	○ Absolute
Fund Unbundling	⦿ Bundled	○ Unbundled

Asset Base for Analysis

Total net Assets	+48,742.00 CHF
Securities flow	+0.00 CHF
Total fictitious assets	+0.00 CHF
Cash flow	+0.00 CHF
Automatically excluded assets	+0.00 CHF
Manually excluded assets	+0.00 CHF
Asset base for analysis	**+48,742.33 CHF**

Asset Allocation vs. Initial Portfolio — Excess/Missing Cash: 0.00 CHF

		Liquidity	Bonds	Equities	Alternative Investments	Mixed/Others	Total
Switzerland	Target AA	7.0	52.0	6.0	20.0		85.0
	Initial	0.0	0.0	61.3	0.0		61.3
	Action	7.0	52.0	-55.3	20.0		23.7
Europe	Target AA		2.5	1.5			4.0
	Initial		0.0	5.5			5.5
	Action		2.5	-4.0			-1.5
Europe non-EUR	Target AA			1.0			1.0
	Initial			0.0			0.0
	Action			1.0			1.0
North America	Target AA		3.5	4.5			8.0
	Initial		0.0	33.2			33.2
	Action		3.5	-28.7			-25.2
Asia/Pacific	Target AA			1.0			1.0
	Initial			0.0			0.0
	Action			1.0			1.0
Japan	Target AA			1.0			1.0
	Initial			0.0			0.0
	Action			1.0			1.0
Mixed/Others	Target AA						
	Initial						
	Action						
Total	Target AA	7.0	58.0	15.0	20.0		100.0
	Initial	0.0	0.0	100.0	0.0		100.0
	Action	7.0	58.0	-85.0	20.0		0.0

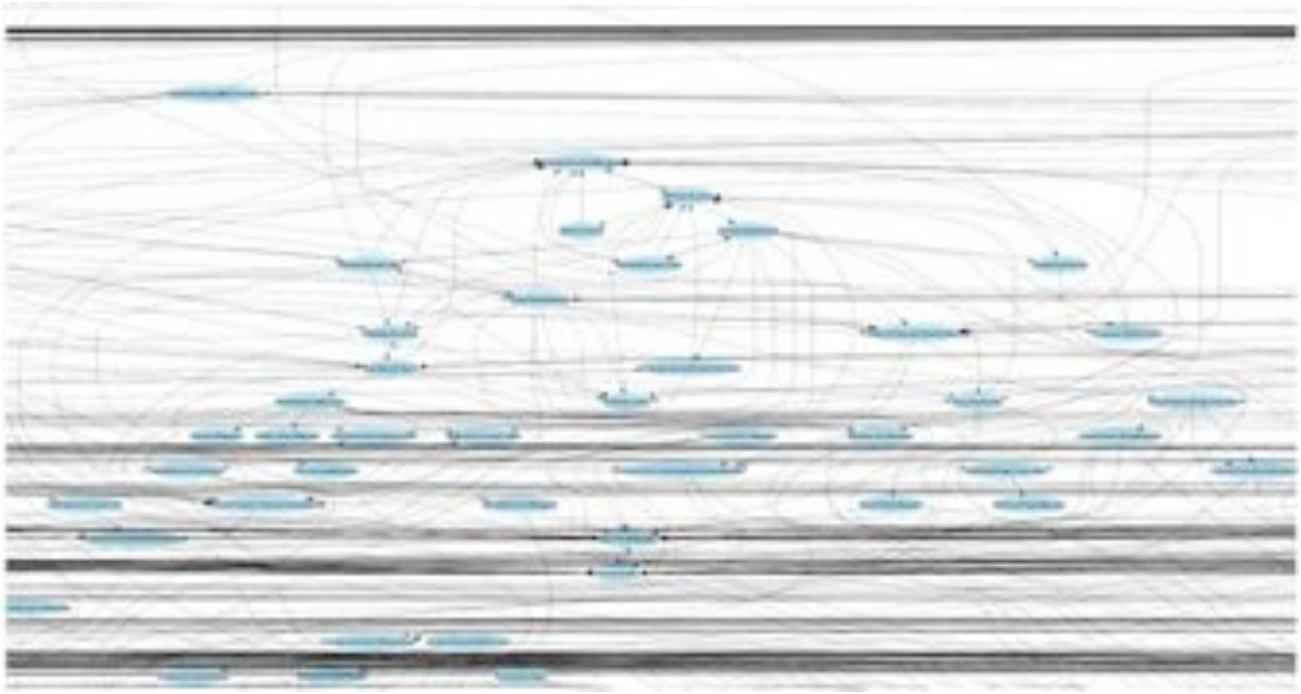
Legend:
- Drill Down to positions and recommendations
- 5.6 buy<6.0%
- 6.0 buy>6.0%
- -5.5 sell<6.0%
- -7.2 sell>6.0%
- switch recommendation
- all switches treated
- TIS settings respected
- TIS settings breached

Facts

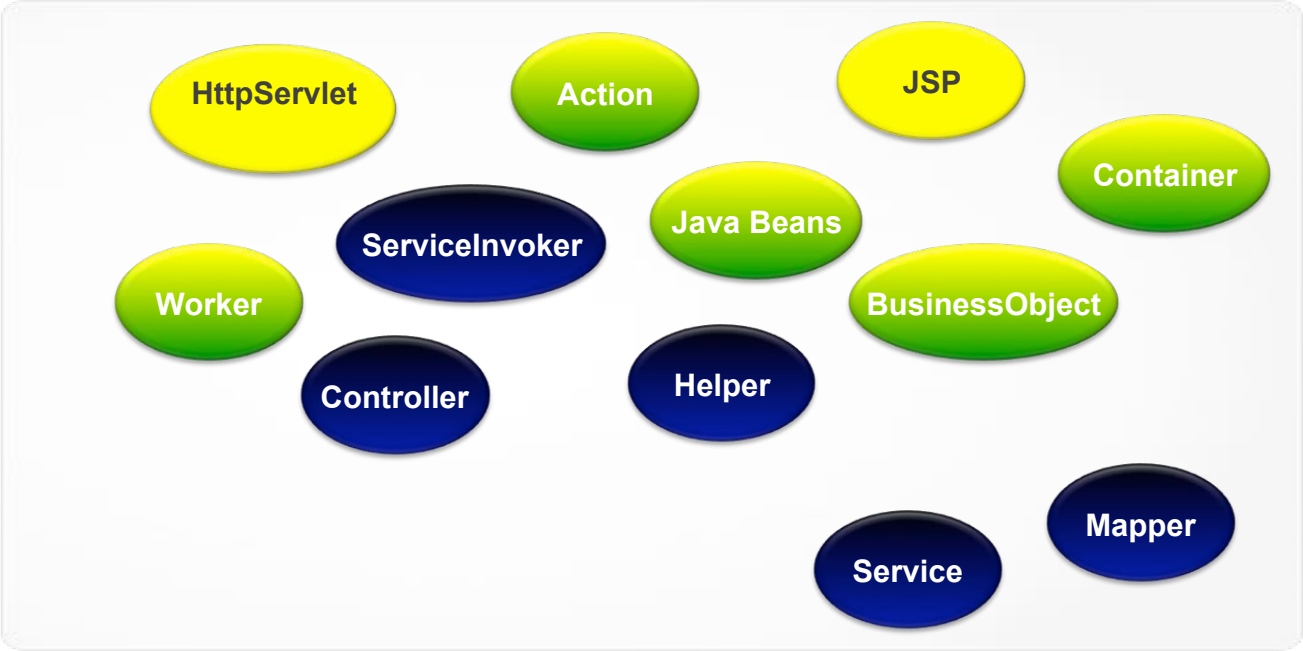
- Produktiv und erfolgreich seit 2004
- Kontinuierlich weiterentwickelt
- 320 JSP Dateien, 1800 Klassen
- Ca 70. funktionale Tests
- Wenig Unit Tests
- Wenig Dokumentation
- Hohe Fluktation, hohe Kosten, Change Requests können nicht mehr implementiert werden.

Start: Erste Analyse

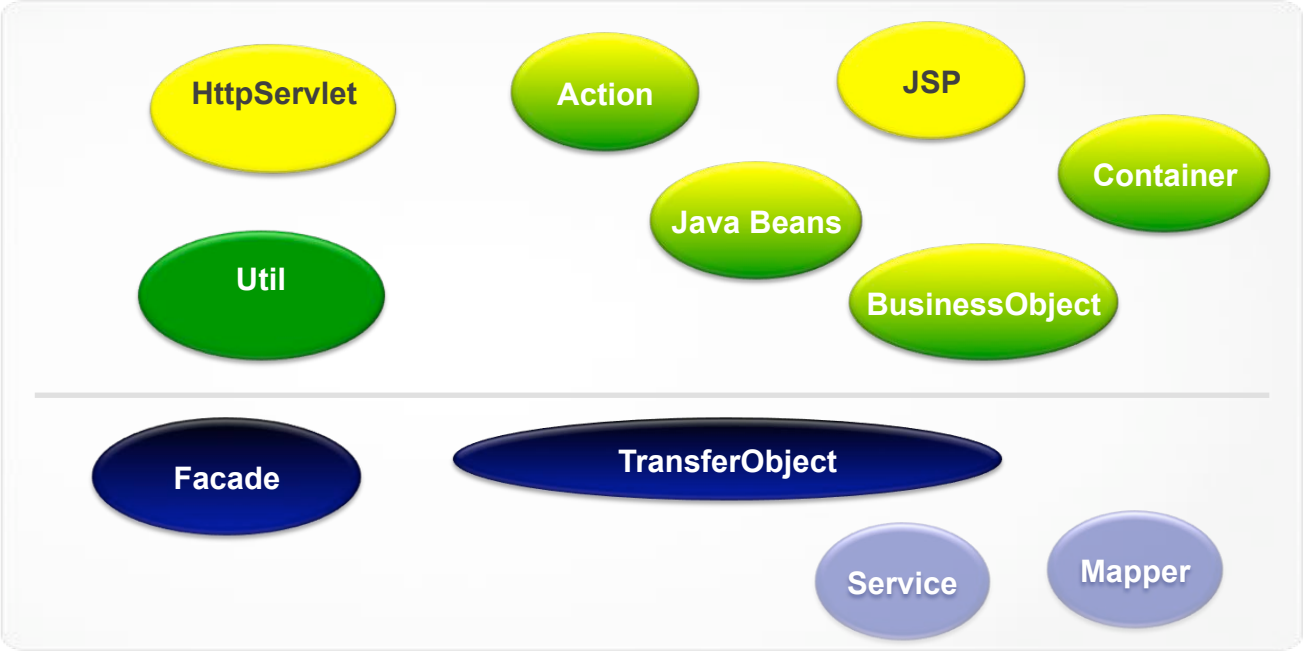
Abhängigkeiten



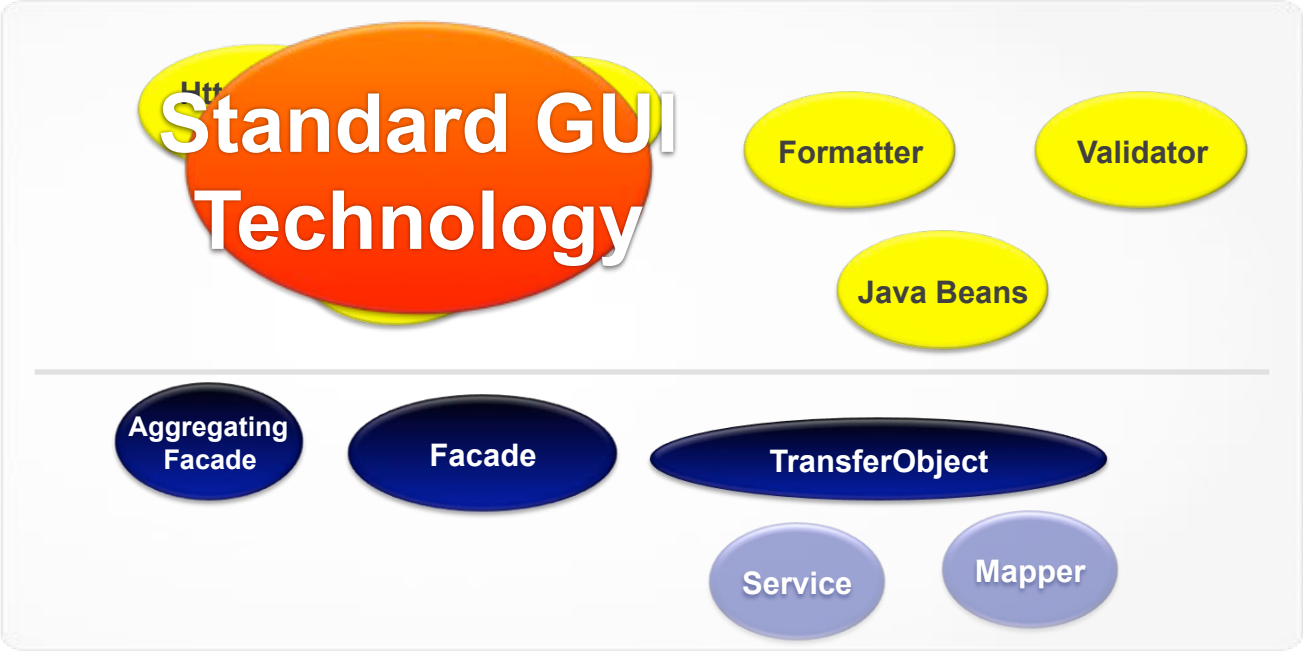
Objekttypen



Masterplan

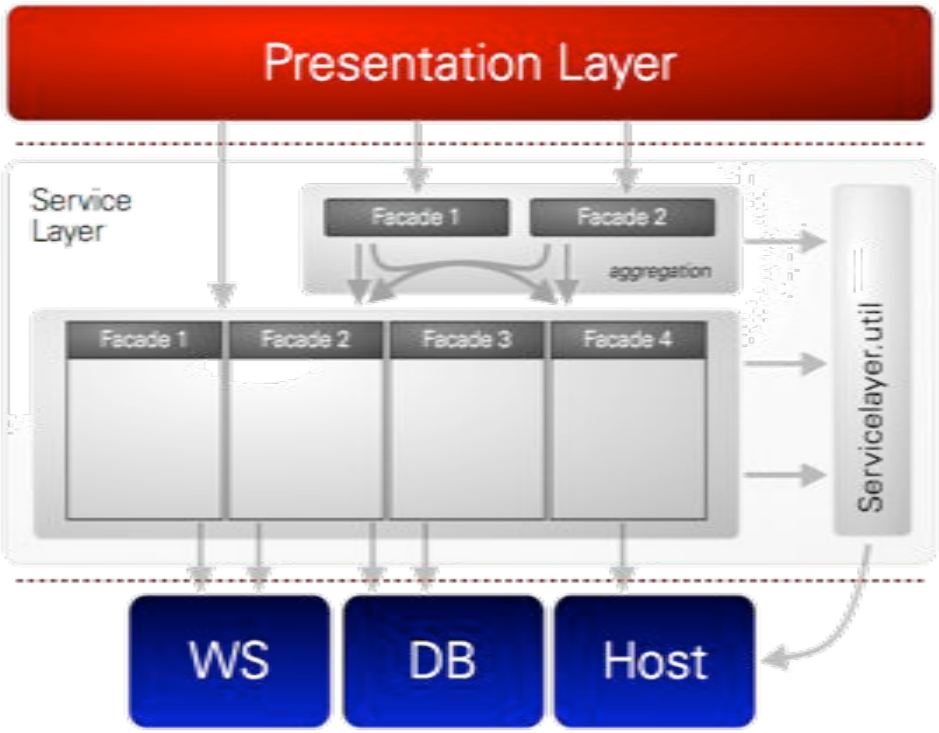

HttpServlet
Action
JSP
Container
Java Beans
Util
BusinessObject
Facade
TransferObject
Service
Mapper

Masterplan

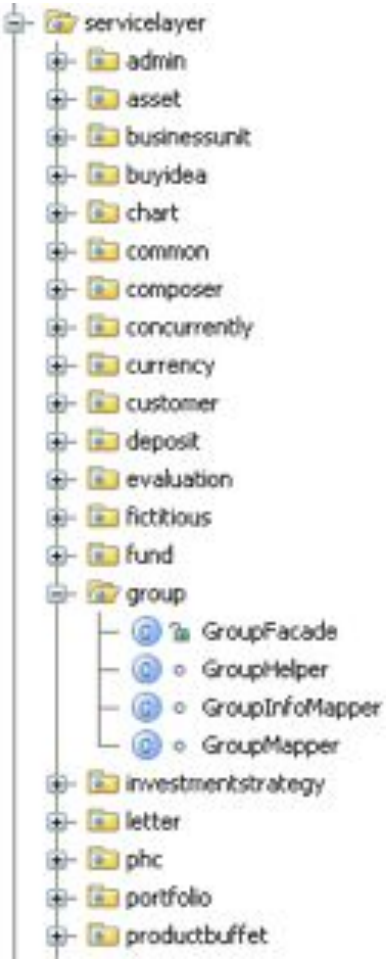

Standard GUI Technology
Formatter
Validator
Java Beans
Aggregating Facade
Facade
TransferObject
Service
Mapper

Refactoring Teil 1

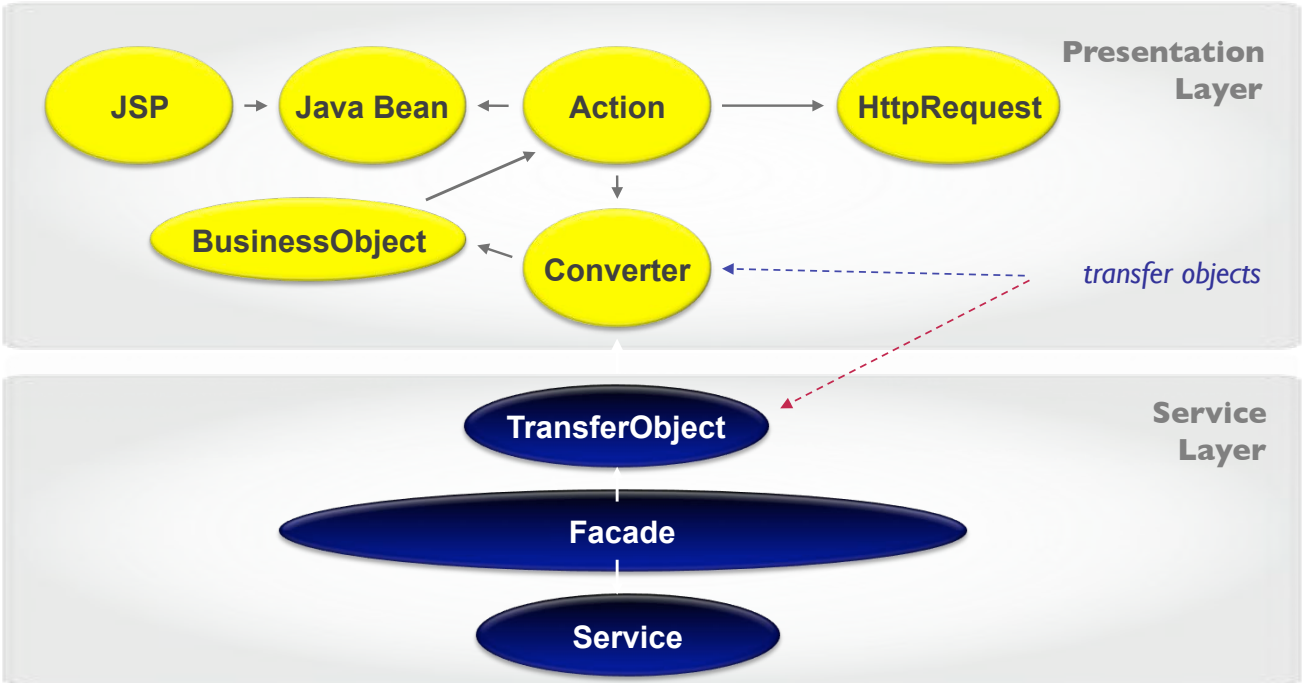
Ziel: Neues Service Layer



1. Schritt Strukturierung in Fassaden

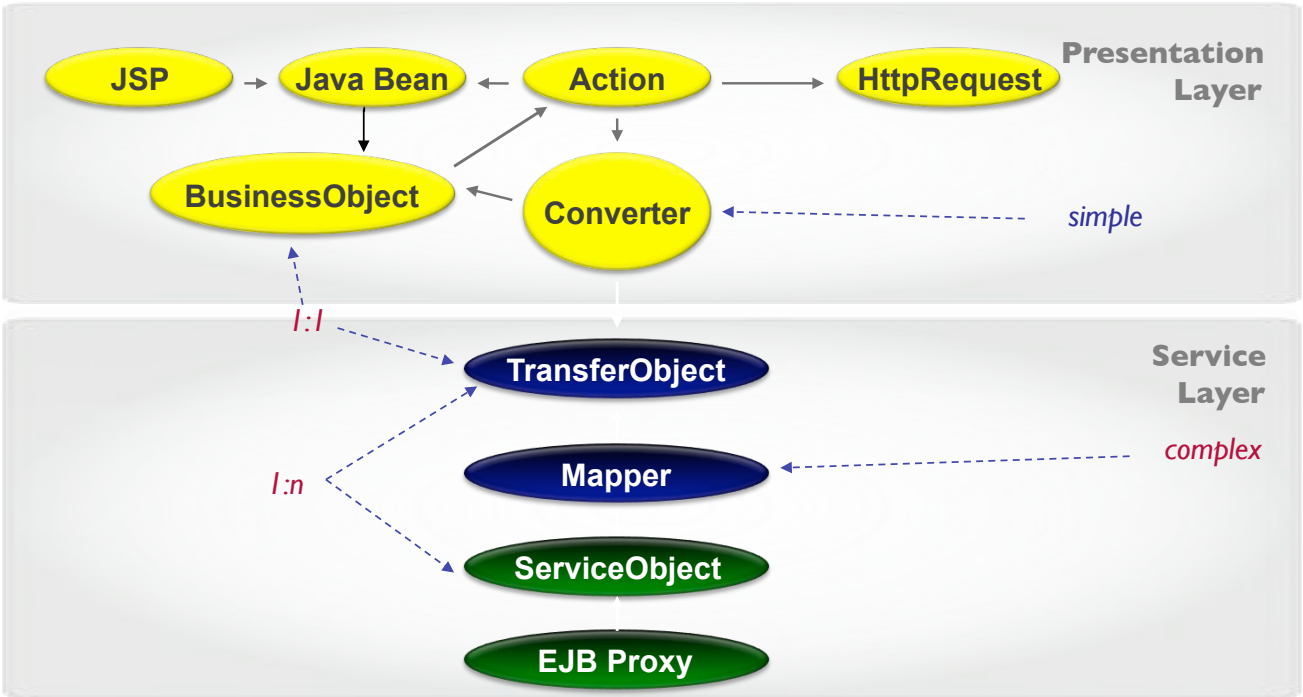


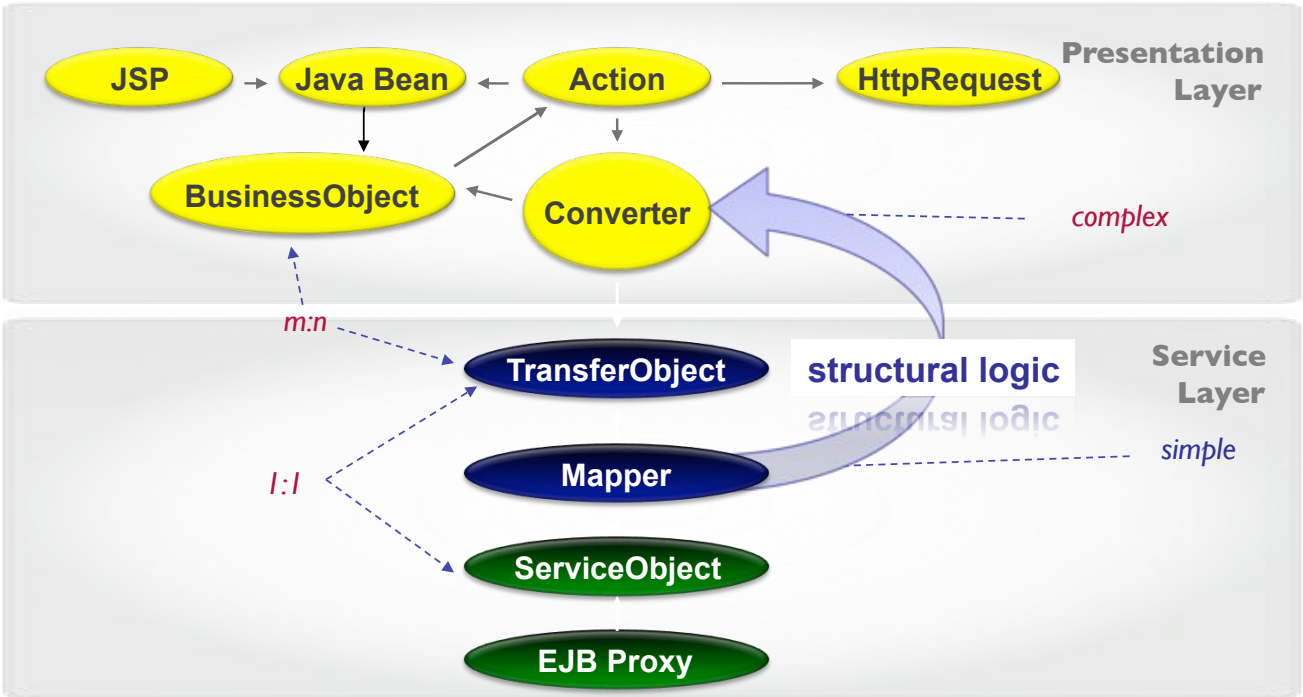
2. Schritt: Einführung von Transfer Objekten (TO)



```
public class User extends CoreObject {
  private String id;
  private Employee userIdentity;
  private RoleBUManager m_roleBUManager = new
RoleBUManager();
  private boolean isAllowed = false;
  private String eMailTask;
  private Timestamp mutTimestamp;
  private String mutId;
```

```
public final class UserTO {
    private final String fId;
    private final String fEamCode;
    private final String fBuCode;
    private final String fDepartment;
```


JSP
Java Bean
Action
HttpRequest
Presentation Layer
BusinessObject
Converter
simple
1:1
TransferObject
Service Layer
complex
1:n
Mapper
ServiceObject
EJB Proxy


JSP
Java Bean
Action
HttpRequest
Presentation Layer
BusinessObject
Converter
complex
m:n
TransferObject
structural logic
Service Layer
Mapper
simple
1:1
ServiceObject
EJB Proxy

Validierung

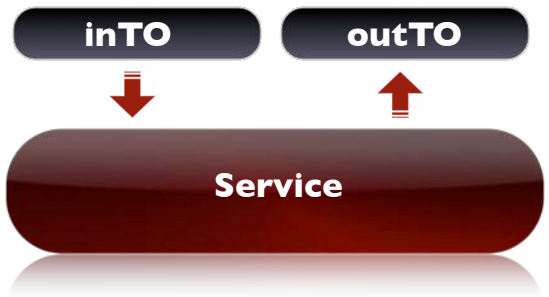
```
public final class AARefCurrencyTO {
  ...
public AARefCurrencyTO(UID<AARefCurrencyTO2> uid,
  String eamCode, final BigDecimal manual, BUCode buCode,
  MutSeqNr mutSeqNr, BigDecimal sortOrder, Long language,
  String name) {
 // Preconditions:
  if (uid == null) throw new IllegalArgumentException(...);
  ...
  if (language > 4) throw new IllegalArgumentException
(...);

  // Assignments:
  this.uid = uid;
  ...
}
```

Fassaden Interface

```
public static FrontpageText getFrontpageText
                          (CriteriaToGetFrontpageText criteria)
                           throws CoreException

public CriteriaToGetFrontpageText(String buCode, User user, int
entryFrom)
```

```
public static FrontpageTextTO getFrontpageText
                                  (String buCode, UserTO user)
                                   throws CoreException
```

3. Schritt: Neues Service Layer

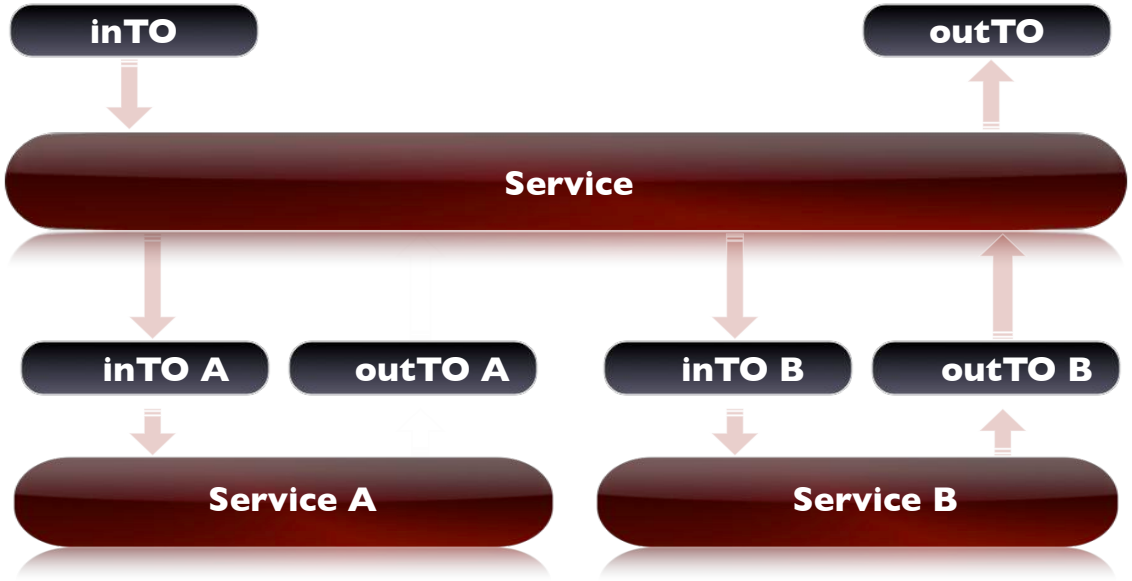


Service Interface und AbstractService

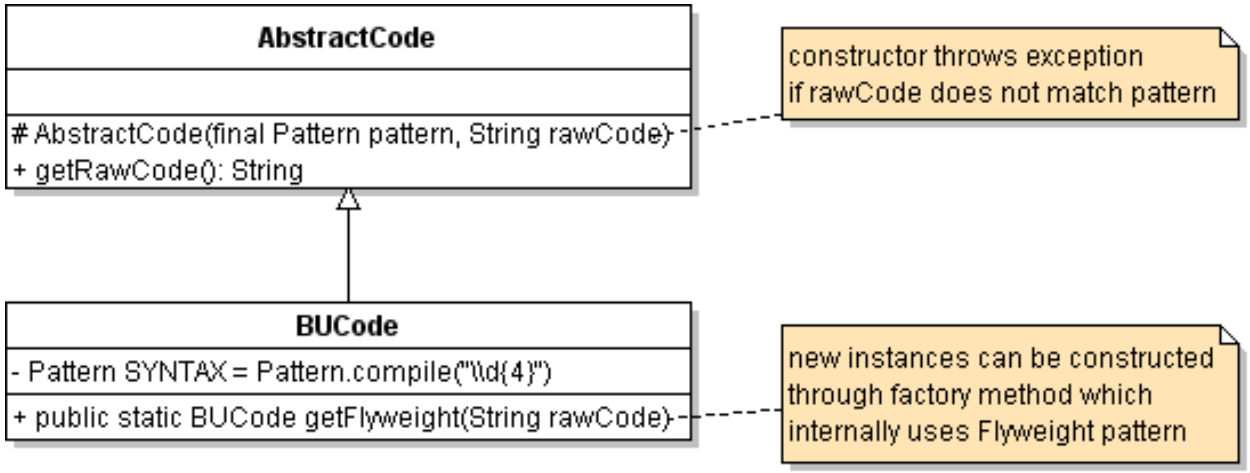
```
public interface IService<TOI, TOO> {
    public TOO execute(TOI toi) throws CoreException;
}
```

```
public abstract class AbstractService<TOI, SOI, SOO, TOO>
implements IService<TOI, TOO> {
    public final TOO execute(final TOI toi)
                                throws CoreException {
      if (toi == null) {
        throw new IllegalArgumentException("'toi' must not
                                            be null");
      }
      SOI soi = mapInput(toi);
      SOO soo = innerExecute(soi);
      return mapOutput(soo);
    }

    protected abstract SOI mapInput(TOI toi);
    protected abstract SOO innerExecute(SOI soi)
                                    throws CoreException;
    protected abstract TOO mapOutput(SOO soo);
}
```

Aggregierende Service Klassen



4. Schritt: Einführung von Basis Typen



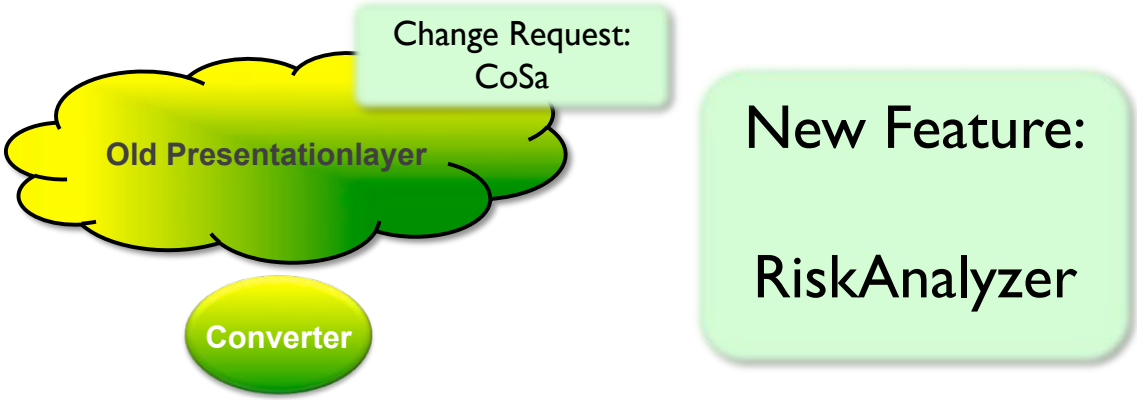

```
// Usage example:
BUCode buCode = BUCode.getFlyweight("0012");
```

Fassaden Interface

```
public static FrontpageTextTO getFrontpageText(String
buCode,
  String eamCode, UserTO user) throws CoreException
```

```
public static FrontpageTextTO getFrontpageText(BUCode
buCode,
  EAMCode eamCode, UserTO user) throws CoreException
```

Resultat nach Teil 1

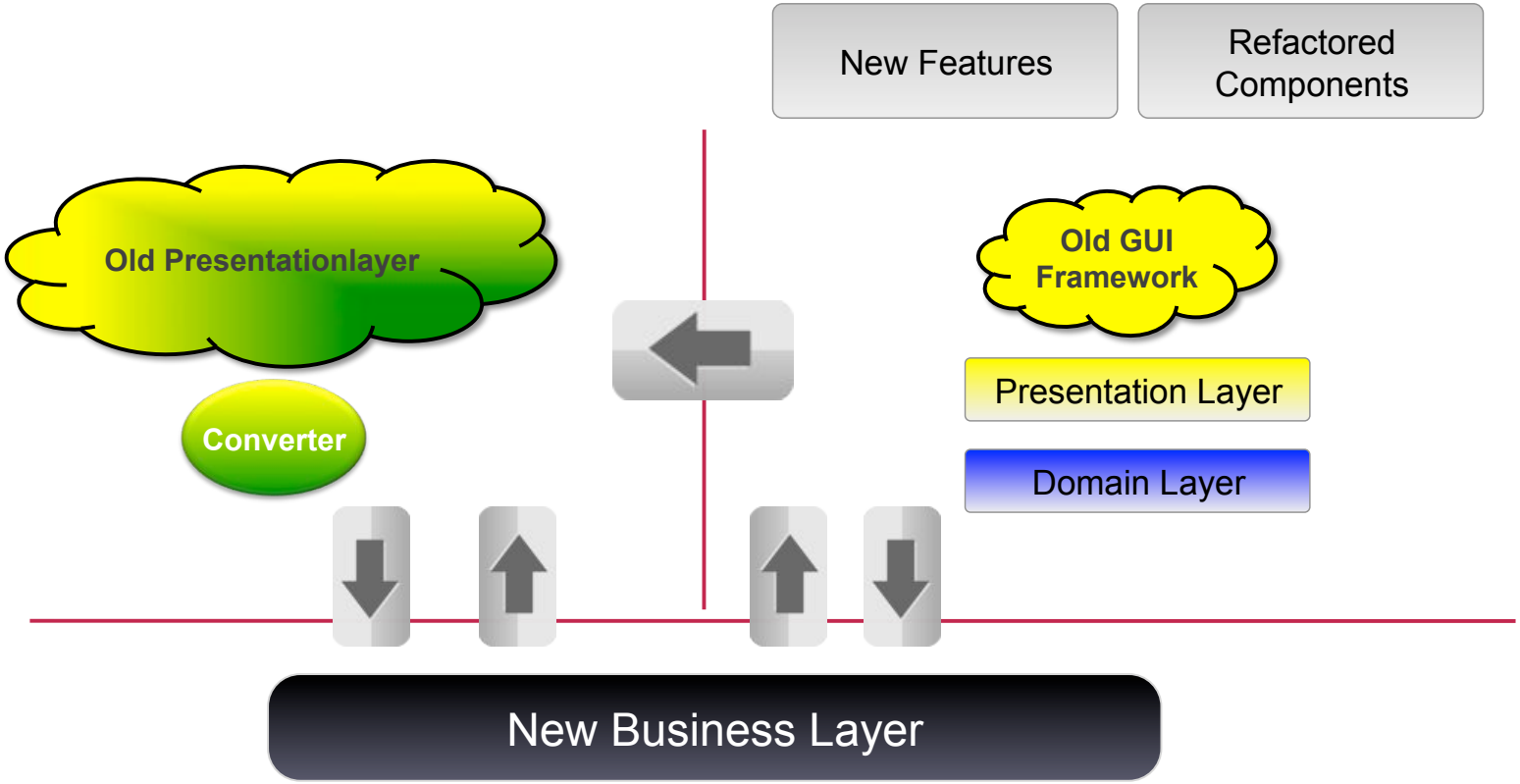


New Servicelayer Architecture

Design Guidelines for
- Services
- TransferObjects
- Basetypes

Refactoring von Komponenten

Current State



Domain Model Adapters

```
public class MonitoredHoldingDO extends AbstractHoldingDO
implements IMonitoredHoldingDO {

    public MonitoredHoldingDO(final Holding holding) {
        super(holding);
    }

    @Override
    protected BOInfo getBOInfo() {
        return new BOInfo(getHolding().getWorkingTIS(),
getAssetsFromHolding              (),
getSafekeepingAccountsFromHolding());
    }

    private ImmutableMap<Integer, Asset>
getAssetsFromHolding() {
        return
HoldingDOBuilderUtil.getTisIncludedAssetsFromHolding
              (getHolding(), getHolding().getWorkingTIS());
    }
    …
}
```

Komponente Load Partner

Vorgehen

- Code aufräumen
  - **IDE Vorschläge** umsetzen (Remove unused code etc)
- Überschaubare Einheiten erstellen
  - Mit **Extract Method** Refactoring Methoden herausziehen
  - Abhängigkeiten zu Attributen entfernen (static methods)
  - **Move** Refactoring benutzen, um diese Methoden in einer Hilfklasse zwischenzulagern

Vorgehen

- **Analyse** der neuen Einheiten und **Design** gemäss der Zielarchitektur
- **Akzeptanz Tests** definieren und mit **BBD Frameworks** implementieren

Akzeptanztest

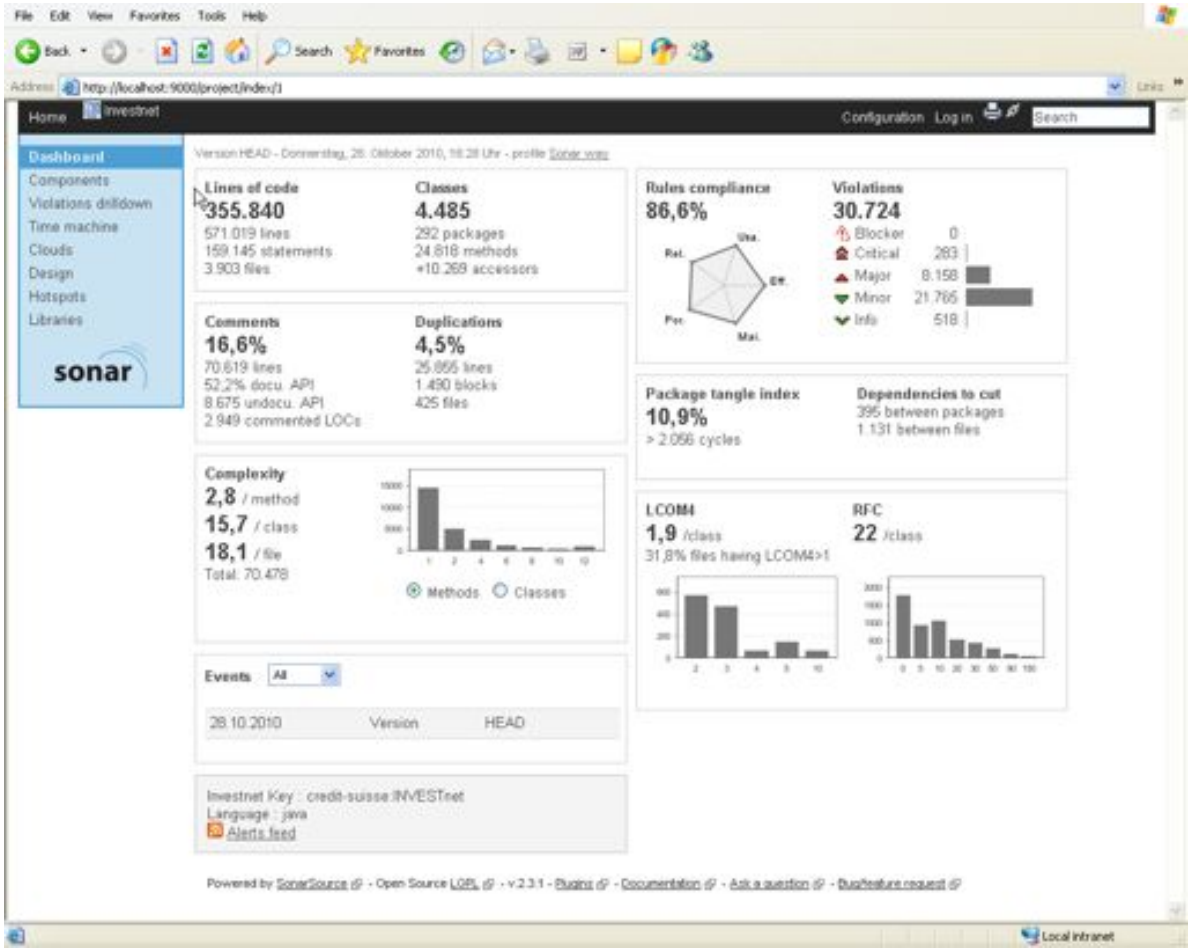
```
scenario "DictionaryService can find the word 'Griffon'", {
   given "an instance of DictionaryService is available", {
       service = new DictionaryService()
   }

   when "the word 'Griffon' is used as parameter", {
       result = service.findDefinition('Griffon')
   }

   then "the definition should be found", {
      assert result == "Grails inspired desktop application
      development platform."
   }
}
```

Akzeptanztest mit EasyB

Vorgehen

- Refactoring
  - Wenn noch nötig, **Mikado Methode** anwenden
- Messen der Code Qualität

Sonar Dashboard



Voraussetzungen für erfolgreiche Sanierungsprojekte?

Wie wir arbeiten

Team

Experienced, dedicated developers

team size.

motivation

A lead engineer of the customers side

A disciplined approach

Authorität und Flexibilität

Management support

Involved in system design

Accepted by the team

The customers trust

Flexible, iterative planning

Weitere Voraussetzungen

Erkenntnisse oder Best Practices

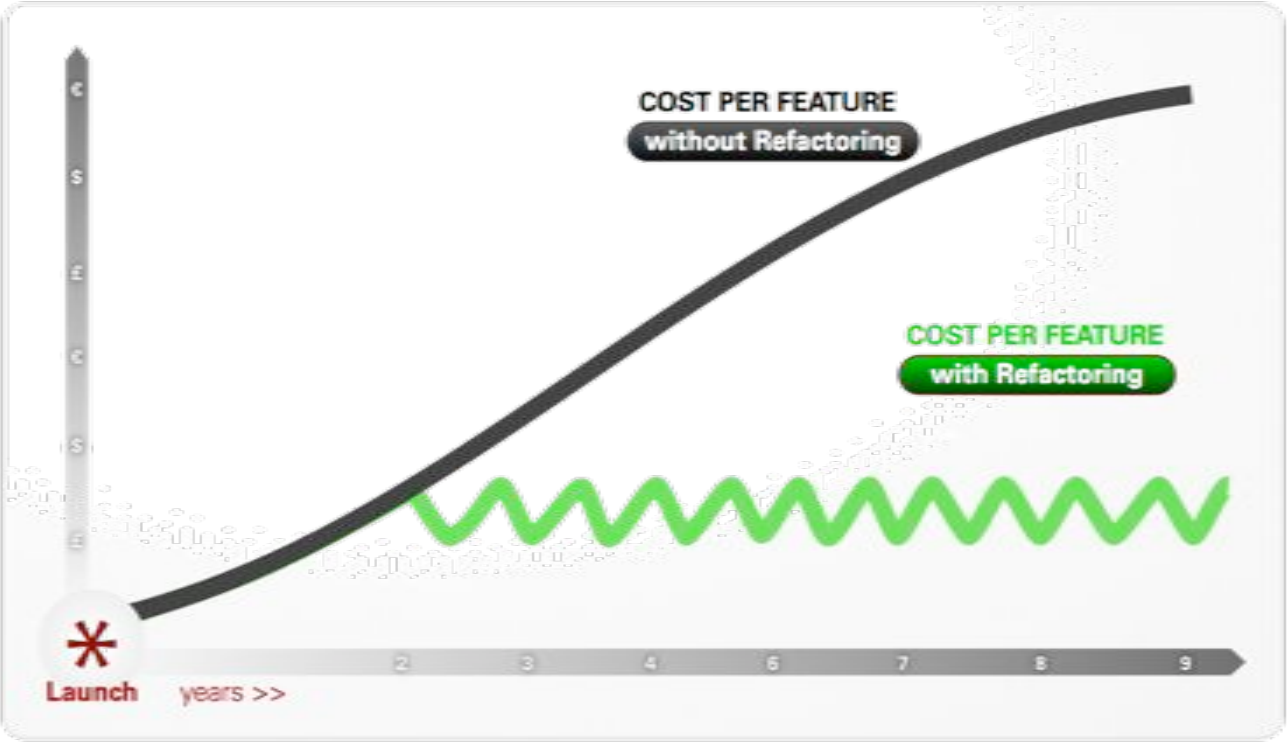
- Entweder technisches oder logisches Refactoring
- Analyse – Implementierung – Stabilisation
- Früher Wissenstransfer zum Core Team
- Investition in automatische Regressionstest
- Abhängigkeiten zu anderen Systemen reduzieren
- Automatische Qualitätssicherung
- Manuelles Testing braucht Zeit – genügend Zeit einplanen

Nachhaltigkeit

- Code verbessern ist nicht genug
- Professionelle Ethik einführen
- Technische Schulden überwachen und ggf. beseitigen
- Continous Improvement: Retrospektiven

- Core Team muss unterstützt werden, damit neue Funktionalität in guter Qualität ausgeliefert wird.

Warum Sanierungsprojekte?



Bücher und Links

- **Martin Fowler**, *Refactoring: Improving the Design of Existing Code*
- **Stephan Ducasse**, **Oscar Nierstrasz**, *Object Oriented Reengineering Patterns*
  PDF: http://www.win.ua.ac.be/~sdemey/
- **Joshua Kerievsky**, *Refactoring to Patterns*
- **Martin Lippert, Stefan Rook**, *Refactoring in Large Software Projects*
- Behaviour Driven Development:
  - EasyB: http://www.easyb.org/
  - Spock: http://code.google.com/p/spock/
- Sonar: http://www.sonarsource.org/
- Mikado Method: http://pragprog.com/magazines/2010-06/the-mikado-method

Master Thesis @ Canoo

Master Thesis @ Canoo