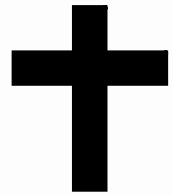
andrena OBJECTS

Refactoring von Legacy Systemen

Jochen Winzen
jochen.winzen@andrena.de
andrena objects ag

Was ist ein Legacy System

- **Ein Legacy System hat folgenden Eigenschaften:**
  - \+ Besitzt die geforderte Funktionalität
  - o Wurde über mehrere Jahre entwickelt
  - \- Ursprüngliche Entwickler nur wenig OO Kenntnisse
  - \- Wenig Know-How vom System
  - \- Testen schwierig
  - \- Jede Änderung kann zu unerwarteten Ergebnissen führen
  - \- Code wird immer unübersichtlicher



Was ist Legacy Code

- **Legacy Code ist *Code ohne Tests***

  → Michael Feathers:
  „Code without tests is bad code. It doesn't matter how well written it is [...] With tests, we can change the behavior of our code quickly and verifiably. Without them, we really don't know if our code is getting better or worse.“

Die Folgen

- edit & pray
- copy & paste
- Der Zustand der Anwendung wird immer schlechter

andrena OBJECTS

Die Zukunft

1. Alles so lassen, wie es ist.
3. **Refactoring des Systems**
2. Neuschreiben

andrena OBJECTS

Refactoring des Systems

- **Änderung des bestehenden Systems**
  - → Neue Features
  - → Entfernen von Bugs
  - → Designverbesserungen
  - → Optimierungen
- **Vergrößerung der Testabdeckung**
  - → Testgetriebene Entwicklung
  - → Akzeptanztests
  - → Integrationstests

andrena OBJECTS

Refactorings

- Umbenennen
- Methode extrahieren
- Schnittstelle extrahieren
- Konstruktor parametrisieren
- Globale Referenzen kapseln
- Extrahieren und Überschreiben
- Statische Setter in Singletons
- Ersetze Parameter durch primitive Typen
- ...

Die ersten Schritte

1. Analyse des Systems (Review)
2. Absichern des bisherigen Funktionsumfangs durch GUI oder Web Tests
3. Abhängigkeiten zu GUI, Web oder System Komponenten aufbrechen
4. Die neuen Schnittstellen durch Akzeptanztest abdecken

- **Achtung: Manchmal wird das Design zuerst ein wenig schlechter**

Wie sichern wir die Refactorings ab

- **Komponententests**
  - → JUnit, NUnit
- **Akzeptanztests**
  - → FIT
  - → GUI Tests
- **Mock Framework**
  - → EasyMock, NMock
- **Refactoring Werkzeuge**
  - → z.B. Refactorings in Entwicklungsumgebung

andrena OBJECTS

Legacy Code Dilemma

- Um Software sicher zu ändern, benötigen wir automatisierte Tests.
- Um automatisierte Tests zu schreiben, müssen wir die Software ändern.

⇨ *Refactoring ohne Tests unumgänglich*

Probleme beim Testen

- **Problem:**
  - → Klasse kann nicht instanziert werden
  - → Hinderliche Konstruktor-Parameter
  - → Konstruktor hat Seiteneffekte
- **Lösung:**
  - → Extrahiere und überschreibe Methode
  - → Extrahiere Interface
  - → Parametrisiere Konstruktor

andrena OBJECTS

Extrahiere und Überschreibe (1)

```
public class A {
  public A() {
    B b = new B();
    b.doStuff();
    ...
  }
}
```

```
public class A {
  public A() {
    B b = makeB();
    b.doStuff();
    ...
  }
  protected B makeB() {
    return new B();
  }
}
```


Extrahiere und Überschreibe (2)

```
public class TestableA extends A {
  protected B makeB() {
    return new DummyB();
  }
}

public class DummyB extends B {
  public void doStuff() {
  }
}
```


Extrahiere Interface

```
public class A {
  public A() {
    B b = makeB();
    b.doStuff();
    ...
  }
  protected B makeB()
  {
    return new B();
  }
}
```

```
public class A {
  public A() {
    IB b = makeB();
    b.doStuff();
    ...
  }
  protected IB makeB()
  {
    return new B();
  }
}
```


andrena OBJECTS

Parametrisiere Konstruktor

```
public class A {
  public A() {
    IB b = makeB();
    b.doStuff();
    ...
  }
}
```

```
public class A {
  private IB b;
  public A(IB b) {
    this.b = b;
    b.doStuff();
    ...
  }
  public A() {
    this(new B());
  }
}
```


Refactorings 1

- Legacy Code enthält oft sehr lange Methoden ›› 100 LOC

→ Extract Method

- Achtung: „Blindes“ Extrahieren verstreut die Logik und verschlimmbessert den Code!

andrena OBJECTS

Refactorings 2

- **Legacy Code hat oft keine Schichten, SQL direkt in GUI**
- → **Isolierung des Datenzugriffs (Extract Method + Facade Pattern)**
  - SQL-Code in Methode einer Facade verschieben
  - Parameter-Objekte einführen

Refactorings 3

- **Legacy Code hat oft keine Kapselung**
- → **Extract Interface**
  - Sobald Facaden stabil, Interface bilden
  - Sobald Parameter-Objekte stabil, Interface bilden

- **Achtung:**
  - → Interfaces enthalten keine „obskuren“ Datentypen!
  - → Immer nur einzelne Refactorings, keine „Großbaustelle“
  - → System ist jederzeit production-stable

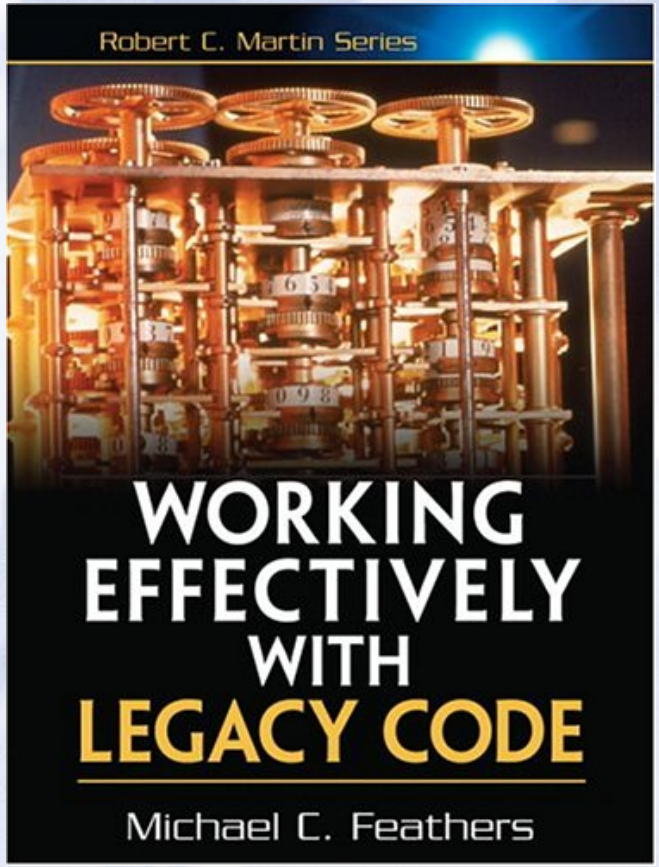
andrena OBJECTS

Fazit

- Refactoring ist eine Lösung für die Probleme mit Legacy Systemen
- Code ohne Tests ist immer problematisch
- Genaue Analyse des Systems ist nötig

andrena OBJECTS

Literatur



Johannes Link,
Frank Westphal,
Legacy Code effektiv weiterentwickeln,
Vortrag w-jax 04

andrena OBJECTS