Refactoring und Patterneinsatz in evolutionärer Umgebung

Seminar Software Engineering :
Agile vs. Klassische Methoden der Software Entwicklung

Jan Krebs

20. Januar 2004

Inhalt

- Evolutionäres Design vs. Geplantes Design
- Refactoring
  - was, warum, wie
- Simple Design und andere Aspekte von Design in XP
- Fazit

Evolutionäres vs. geplantes Design

Evolutionäres Design:

- Design wächst während der Implementation
- Aggregation von taktischen ad-hoc Entscheidungen
- Entropie

Geplantes Design:

- Ingenieursdisziplin
- Architekten vs. Programmierer
- Kann man im Vornherein schon alles wissen?
- Wie reagiert man auf ändernde Anforderungen?

Design bei agilen Methoden

- Evolutionärer Ansatz
- Kostenkurve für Änderungen --> flach
  - durch *enabling practices*
- Wichtig:
  - Testen
  - kontinuierliche Integration
  - Refactoring

Refactoring(1)

- Definition:
  - Refactoring(noun): a change made to the internal structure of a software to make it easier to understand and cheaper to modify without changing ist observable behaviour [Fowler00]
  - Refactoring(verb): to restructure software by applying a series of refactorings without changing ist observable behaviour.[Fowler00]

Refactoring(3)

- Ziel:
  - Software nachher:
    - einfacher zu verstehen
    - einfacher zu ändern
  - Refactoring vs. Performance-Optimierung
    - beides verändert die Struktur, ohne die Funktionalität zu verändern.

Refactoring(5)

- Anlass:
  - Hinzufügen einer neuen Funktionalität
  - Fehlerbehebung
  - Code Review --> Pair-Programming
- Vorgehensweise:
  - Die 2 Hüte
    - Hinzufügen neuer Funktionalität
    - Refactoring
  - Testen nach jedem Schritt

Refactoring(6)

- Ändern wenn es stinkt:
  - „If it stinks, change it. -Grandma Beck discussing child-rearing philosophy“ [Fowler00]
  - „Handfeste Kriterien“ vs. „unästhetisches Design“
  - Keine Metriken
  - Katalog von Code Strukturen --> Bad Smells:
    - Duplizierter Code
    - Lange Methoden
    - Switch Statements
    - usw.

Refactoring(7)

- Refactoring-Bibel von Martin Fowler
- Musterkatalog mit über 70 einzelnen Refactorings beschrieben mit
  - Name
  - Zusammenfassung
  - Motivation
  - Vorgehensweise
  - Beispiel

Refactoring(8)

- Wann soll man kein Refactoring machen?
  - Wenn der Code so schlecht ist, dass es schneller gehen würde, nochmals ganz von vorne anzufangen.
  - Wenn man kurz vor einem Abgabetermin steht.

Simple Design

- „You Ain‘t Gonna Need It“ =YAGNI
- „Do the Simplest Thing that Could Possibly Work“
- Frameworks, wiederverwendbare Komponenten, flexibles Design und Entwurfsmuster.....verboten?

Simple Design - Motivation

- Ökonomie
  - Arbeite ich heute an etwas, was es erst in Zukunft geben muss, dann erhöhe ich das Risiko, das nicht fertigzustellen, was ich heute erledigen muss.
  - Noch schlimmer als an etwas zu früh zu arbeiten ist, es auch noch falsch zu machen
- Komplexität

Verletzt Refactoring YAGNI?

- Refactoring kostet Zeit, doch es wird keine neue Funktionalität implementiert.
- YAGNI: Kein komplexeres Design als nötig, um die aktuelle Story zu implementieren
- Refactoring wird benötigt, um das Design so einfach zu möglich halten.

Architektur in XP

- XP verzichtet weitgehend auf Architektur-Dokument zu Projektstart
- So einfach wie möglich....aber gibt es nicht Sachen, die im Nachhinein schmerzlich hinzuzufügen sind ?
- Fowler plädiert für eine Grobarchitektur --> aber nicht in Stein gemeisselt....

Wer macht das Design in XP?

- Design „entsteht“ während dem Implementieren durch Refactoring
- Jeder Programmierer trägt seinen Teil dazu bei

- Bleiben die Architekten auf der Strecke?

Patterneinsatz in XP?

- Entwurfsmuster-Bibel der „Gang of Four“
- Vorreiter von XP ≈ Vorreiter der Pattern-Bewegung ?
- XP sagt:
  - Gebrauche kein Entwurfsmuster bis Du es benötigst.
  - Finde Deinen Weg in ein Entwurfsmuster über eine einfache Implementation.
- Theorie: Die Kräfte von Simple Design führen einem automatisch zu den Entwurfsmustern.

Fazit

- Ist Design tot?
  - Nein, Design hat nur ein neues Verständnis gefunden.
  - Gutes Design ist überaus wichtig bei agilen Methoden, denn ohne gutes Design sind späte Änderungen ein Greuel.
- Refactoring
  - eine disziplinierte Methode zur Verbesserung des Designs von existierendem Code

Diskussion