Karlsruher Entwicklertag 2013

Coding Dojos im Unternehmen

...und es lohnt sich doch

05.06.2013

Agenda

- Vorstellung
- Vorgeschichte
- Basics
- Beginn
- Clean Code & TDD
- Fazit

Ralf Schoch – Zur Person

- Freiberuflicher IT-Consultant
- Diplom-Betriebswirt (WI)
- Certified Scrum Product Owner (CSPO)
- Projektmanagement, Coach, Entwickler
- Softwareentwicklung seit 1992 (IBM3090)
- MS Technologie (Web/PC/...) seit 1998
- Schwerpunkt: .Net, C#, Scrum, CCD
- .Net Open Space Süd, .NET User Group Karlsruhe. Karlsruher Coding Dojo, VKSI
- Karate Trainer

Agenda

- ✓ Vorstellung
- Vorgeschichte
- Basics
- Beginn
- Clean Code & TDD
- Fazit

Test Driven Development – First Contact

- 2009 auf dem Open Space in Leipzig beim Coding Dojo mit Ralf Westphal und Stefan Lieser
- Januar 2010 - Erstes Coding Dojo bei der .Net User Group Karlsruhe
- Mai 2010 – April 2012 als Entwickler in einem Scrum Team mit TDD & Pair Programming
- Wöchentliches Coding Dojo im Team
- Monatliches Coding Dojo in der User Group Karlsruhe

Clean Code – First Contact

- Ende 2008 als Projektmanager in einem Projekt mit „schlechter“ Code Qualität
- Vortrag von Stefan Lieser bei der .Net User Group Karlsruhe im April 2009
  - „Clean Code“ Buch von Robert C. Martin (Uncle Bob)
  - Professioneller Softwareentwickler
  - Prinzipien
  - Praktiken
  - „Clean Code Developer“ Armbänder
  - Bewusstsein

Vorgeschichte

- Neues Projekt
  - Projektmanager statt Entwickler ☹
  - Ziel Schadensbegrenzung
  - Erster Kontakt mit SCRUM
  - Viel hilft viel: Teamgröße 2 -> 10
  - Grundproblem: Schlechte Code Qualität
- Clean-Code-Developer Vortrag (Stefan Lieser)
  - Erster Kontakt mit Clean Code

Vorgeschichte

- .Net Open Space Leipzig
  - Erstes Coding Dojo (Ralf Westphal, Stefan Lieser)
  - 150 Teilnehmer
  - Chaotischer Eindruck
- .Net User Group Karlsruhe
  - Reine Vorträge sind zu theoretisch
  - Wunsch nach mehr coden
  - Erstes Coding Dojo in der User Group
  - Später monatliche Dojos

Vorgeschichte

- Neues Projekt
  - Als „normaler“ Entwickler in einem Scrum Team
  - Auf der grünen Wiese anfangen (Alt-SW ablösen)
  - Scrum konsequent umgesetzt
  - Agile Entwicklunsmethoden
    - Pair Programming
    - Test Driven Development
  - Wöchentliche Coding Dojos
  - 2 Jahre lang

Vorgeschichte

- Angebot: Altes Projekt, neue Rolle
  - Entwicklungsleiter
  - Alte Codebasis (~4 Jahre)
  - Bestehendes Team, aber alles Einzelkämpfer
    - Code Ownership (Module)
    - Individueller Stil
    - Jedes Modul anderst
    - Unterschiedliche Auffassung von Qualität
    - Mehrere Mitarbeiter haben Projekt verlassen
    - Aufwändige Weiterentwicklung
    - Viele Bugs, etc.

Vorgeschichte

- Ziele
  - Qualität verbessern
  - Weniger Bugs
  - Bessere Skalierung im Team
  - Ausfallsicherheit
  - Wissenstransfer
  - Gemeinsames Codeverständnis
- Wie umsetzen?
  - Ein Mittel: Coding Dojos

Agenda

- ✓ Vorstellung
- ✓ Vorgeschichte
- Basics
- Beginn
- Clean Code & TDD
- Fazit

Basics

- Warum wollen wir denn überhaupt ein Coding Dojo machen?
- Was ist ein Dojo?
- Was ist eine Kata?

Warum Dojos?

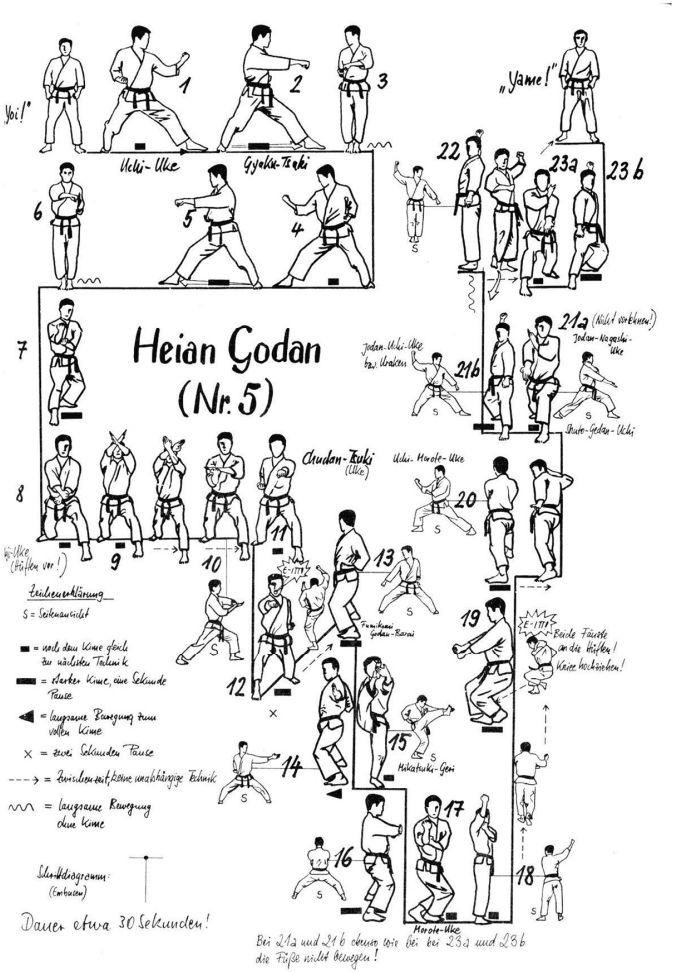
- Teambildung.
- Gemeinsames lernen.
- Gemeinsames erfahren.
- Miteinander kommunizieren.

Was ist ein Dojo?

- Trainingsraum.
- Kommt aus den japanischen Kampfsportarten.
- Raum zum Lernen und Lehren.
- Schutz vor äußeren Einflüssen.
- Aneignen von neuen Praktiken.
- Eigene Fähigkeiten verbessern.
- Coding Dojo

Was ist eine Kata?

- Kommt aus den japanischen Kampfsportarten.
- Klar definierte Aufgabe.
- Interpretationsspielraum bei Umsetzung.
- Wiederholbar.
- Man spricht über das gleiche.
- Coding Kata



Agenda

- ✓ Vorstellung
- ✓ Vorgeschichte
- ✓ Basics
- Beginn
- Clean Code & TDD
- Fazit

Wie beginnen ...

- Nicht anfangen über die Vorteile zu diskutieren!
- Kleine, überschaubare Aufgabe zum Lösen geben.
- Selbst diese Aufgabe systematisch & reproduzierbar mit Babysteps lösen.

Danach

- Das Ergebnis für sich sprechen lassen.
- Anhand der konkreten Beispiele Vor- und Nachteile diskutieren.

Überzeugungsarbeit

- Kleines Experiment
- 2-3 Entwickler
- Dauer ca. 2 Stunden
  - 1. Stunde
  - Jeder darf entwickeln wie er möchte
  - Ergebnis vergleichen
  - 2. Stunde
  - Gemeinsam mit Test Driven Development (TDD)
- Bsp.: Kata Prime Factors

Ergebnis

- Code ist sehr unterschiedlich.
- Code ist umfangreich (Klassen, Methoden, Umfang).
- Code ist (teilweise) komplex.
- Es ist schwierig den Code der anderen zu verstehen.
- …

Akzeptanz Probleme?

- Unser Management erlaubt uns kein TDD!
- TDD kostet mehr Zeit.
- Dojos & Katas haben nicht mit der Realität zu tun.
- Das funktioniert nicht mit unserem Code.
- Falsche Erwartungshaltung (Der Weg ist das Ziel).
- ...

Agenda

- ✓ Vorstellung
- ✓ Vorgeschichte
- ✓ Basics
- ✓ Beginn
- Clean Code & TDD
- Fazit

Coding Dojo - Modi

Klassiker:

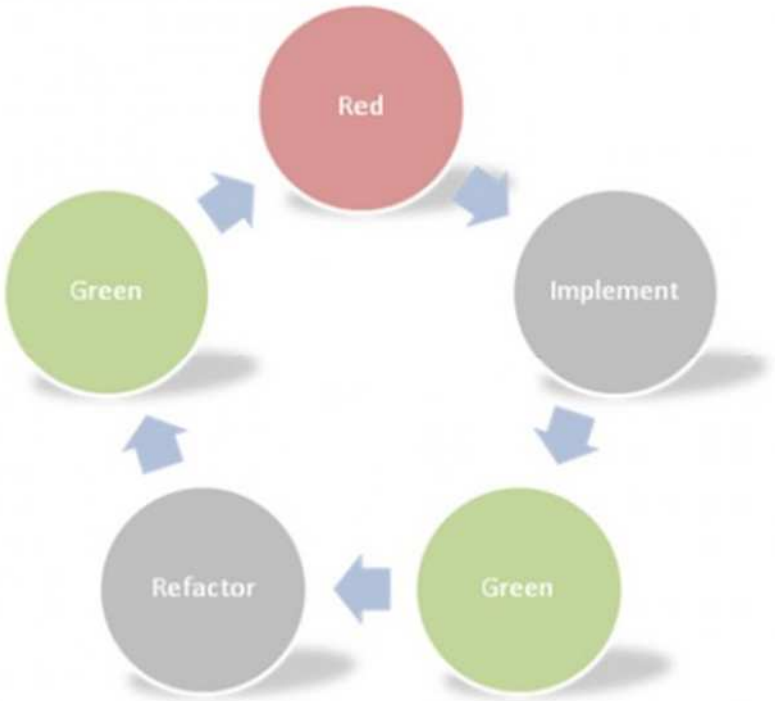
- Prepari Modus
  - Code Monkey
- Randori Modus
  - Pilot / Co-Pilot

Wie wir es machen:

- Kaeshi-Ippon (aka Ping-Pong)
  - TDD-Zyklus

TDD - Test Driven Development ...

TDD Mantra (Red-Green-Refactor)



TDD - Test Driven Development ...

Merkmale von TDD

- Erst Test, dann Code
  - Keine Zeile Produktive-Code ohne Test
  - Für jeden Code muss es eine Begründung in Form eines Tests geben
  - Code ist immer testbar
  - Code wird in ausführbarer Weise dokumentiert
  - Tests dokumentieren die Verwendung des Codes
- Fehler mit Test dokumentieren, dann beheben
  - Änderung im Test dokumentieren Fehler oder geänderte Anforderungen
  - Änderungen an Anforderungen werden eingecheckt

TDD - Test Driven Development ...

Auswirkungen von TDD

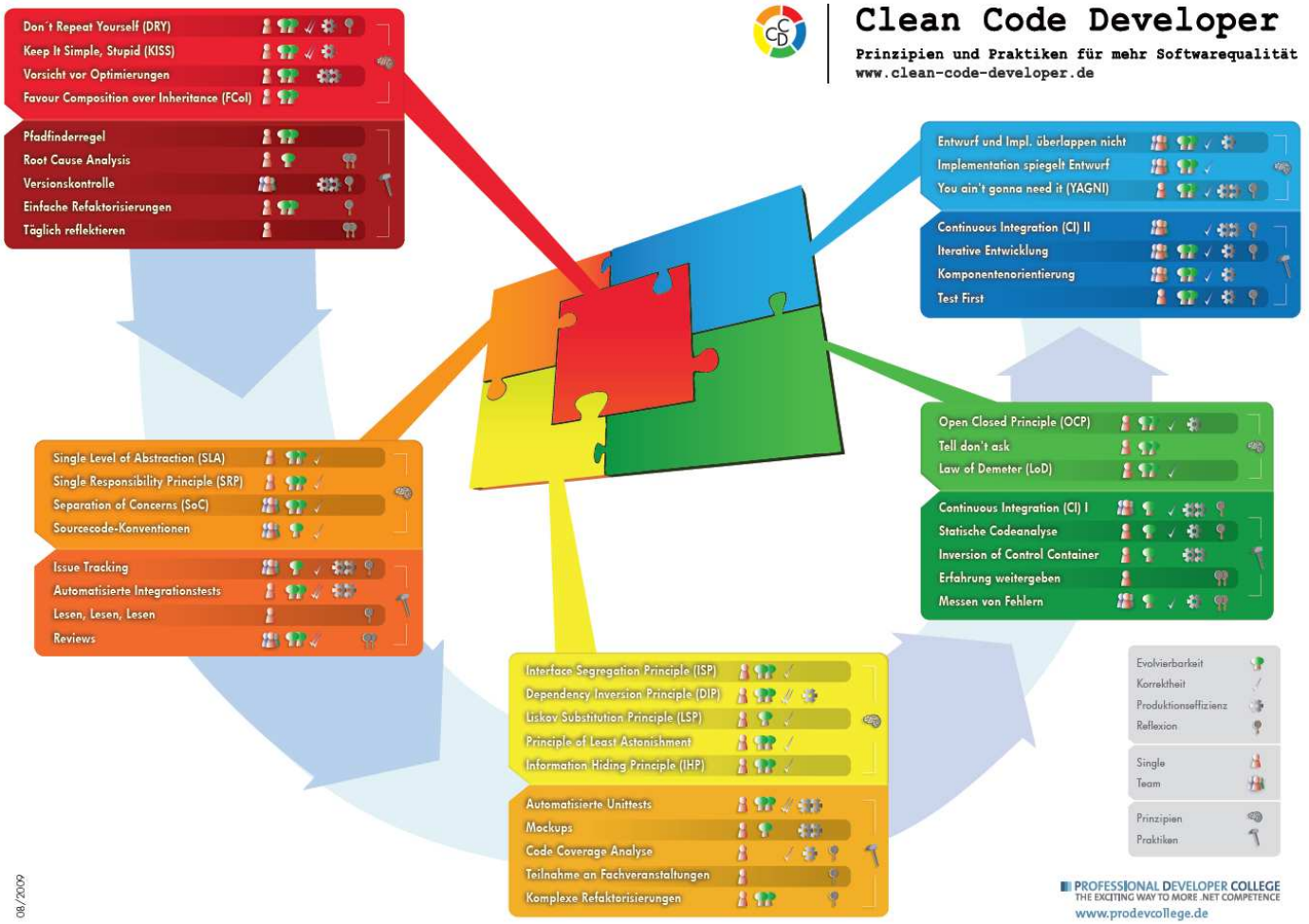
- Entwickler denken mehr über Anforderungen nach
- Nicht-Kunden Anforderungen werden dokumentiert
- Entwickler Doku passt immer zum Code
- Refactoren ist tägliche Praxis
- Code wird leichter erweiterbar
- Losere Kopplung von Komponenten
- Fehler/Unstimmigkeiten werden früher entdeckt

… oder Test Driven Design?

Änderung der Perspektive

- Top-Down-Ansatz
- Kundenlösung steht im Vordergrund
- Lösungs-, nicht Technikorientiert
- KISS
- Ergänzt Scrum Prozess

Clean Code Developer (CCD)



CCD - Prinzipien

Beispiele

- Don‘t Repeat Yourself (DRY)
- Keep It Simple, Stupid (KISS)
- Single Level of Abstraction (SLA)
- You ain‘t gonna need it (YAGNI)
- Open Closed Principle (OCP)

CCD - Praktiken

Beispiele

- Pfadfinderregel
- Versionskontrolle
- Täglich reflektieren
- Lesen, Lesen, Lesen
- Automatische Unittests
- Test First

Agenda

- ✓ Vorstellung
- ✓ Vorgeschichte
- ✓ Basics
- ✓ Beginn
- ✓ Clean Code & TDD
- Fazit

Fazit

Was hat sich für das Team verändert?

- Team kommuniziert heute auf einem anderen Level
- Die Performance ist ungefähr gleich geblieben, obwohl inzwischen zwei Personen weniger
- Keine Einzelkämpfer mehr
- Entwickeln ist entspannter geworden
- Der neue Code ist einfacher geworden
- Er ist leichter erweiterbar geworden
- Es werden mehr Tests geschrieben
- Mehr Zeit für Weiterbildung

Fazit

Clean Code

- Fundament
- Lebenslanges Lernen
- Stetiger Verbesserungsprozess

Test Driven Development

- Ein Rahmen für Entwickler
- Praktische Umsetzung
- Sicherstellung von Qualität

Fazit

Coding Dojos

- Rahmen zum Lernen
- Wissenstransfer
- Kommunikation
- Gemeinsames Verständnis
- Klare Zielsetzung für Übung
- Immer Reflektieren (Retrospektive)
- Regelmässiges trainieren
- Gelerntes fliesst in Praxis ein. Nicht erzwingen!

Quellen

Coding Dojo

http://www.codingdojo.org/

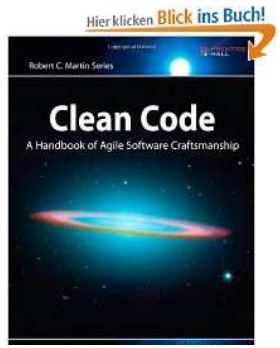
Literatur

Clean Code Developer

http://www.clean-code-developer.de/

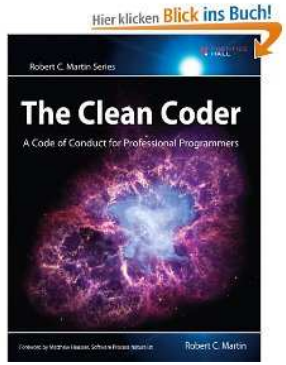
Clean Code – Robert C. Martin

ISBN 0132350882



The Clean Coder – Robert C. Martin

ISBN 0137081073



Fragen

Werbung

.Net Open Space Süd (6.-7. Juli 2013)

http://karlsruhe.netopenspace.de/2013/

.Net User Group Karlsruhe

http://www.dotnet-ka.de/

Kontakt: Ralf Schoch

mail ralf.schoch@codeso.de

xing https://www.xing.com/profile/Ralf_Schoch