Die

Initiative (CCD)

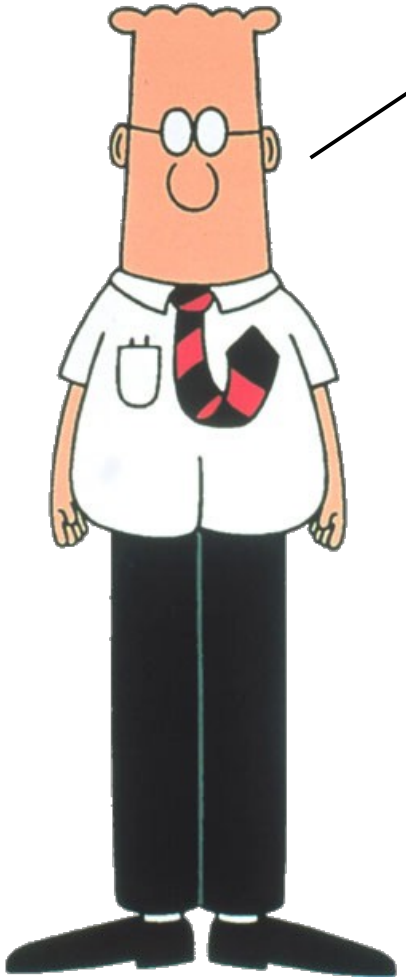
DHBW

Softwareentwicklung

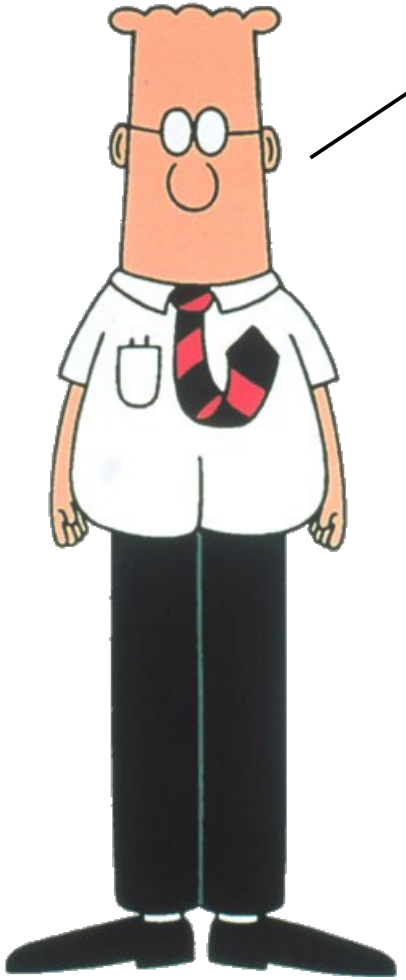
Softwareentwicklung wird von Menschen ausgeführt

DHBW

Softwareentwicklung



Softwareentwicklung

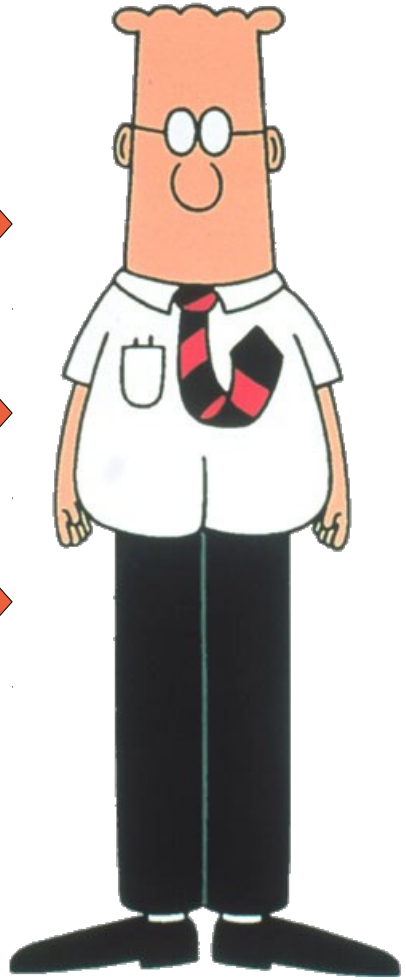


Der professionelle Entwickler

Softwareentwicklung

Äußere Einflüße

- Termindruck
- Budgetzwang
- Qualitäts-anforderung



DHBW

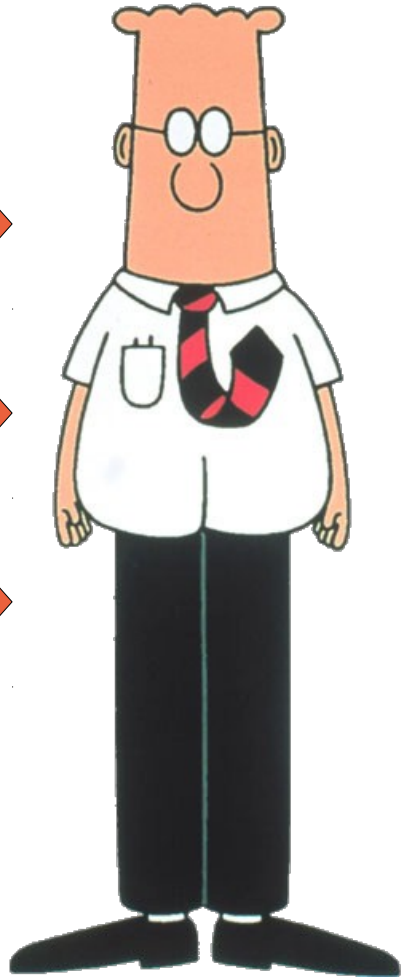
Projektdreieck

DHBW

Softwareentwicklung

Äußere Einflüße

- Termindruck
- Budgetzwang
- Qualitäts-anforderung

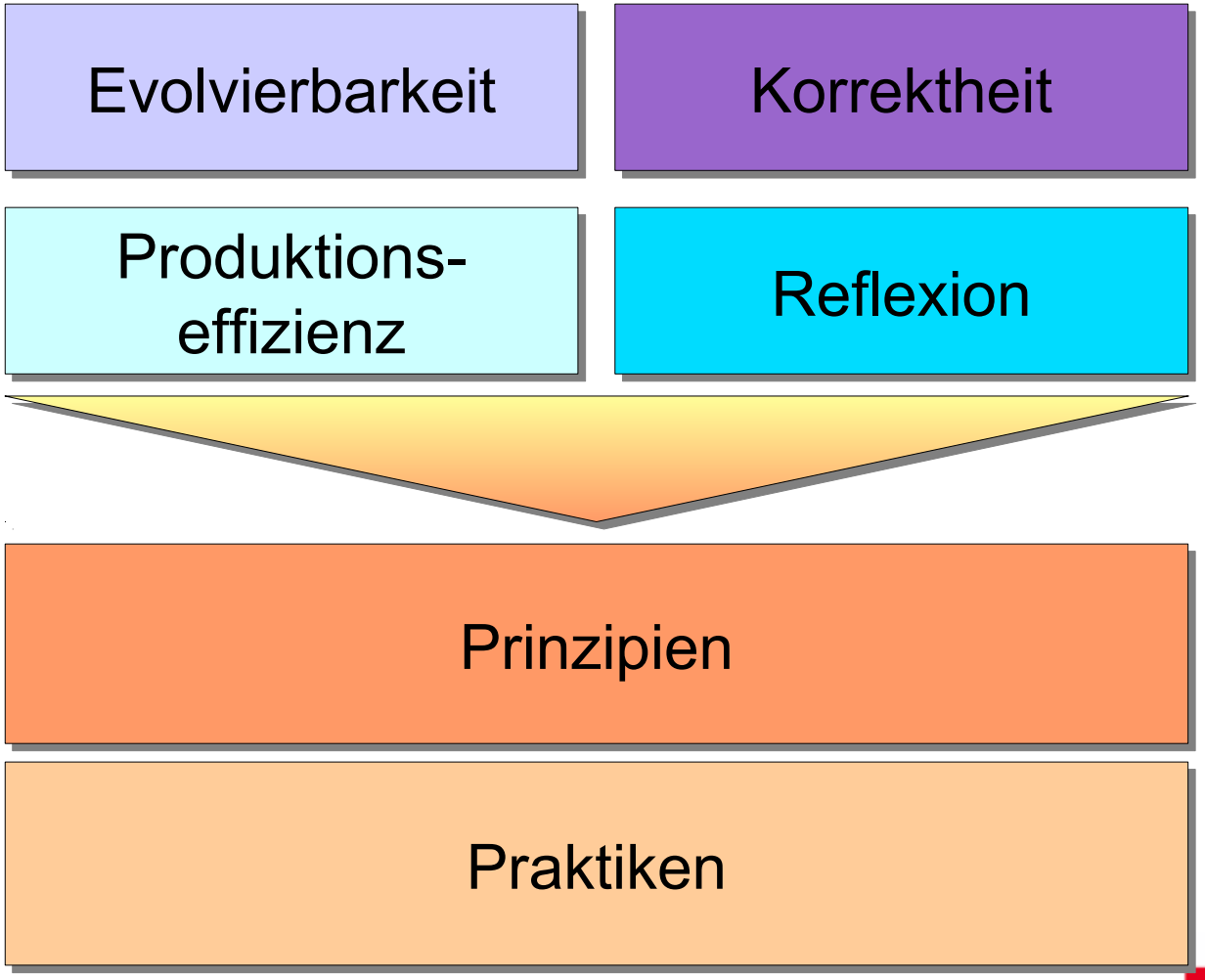


Innere Einflüße

- Wissen & Erfahrung
- Prozess
- Werte

DHBW

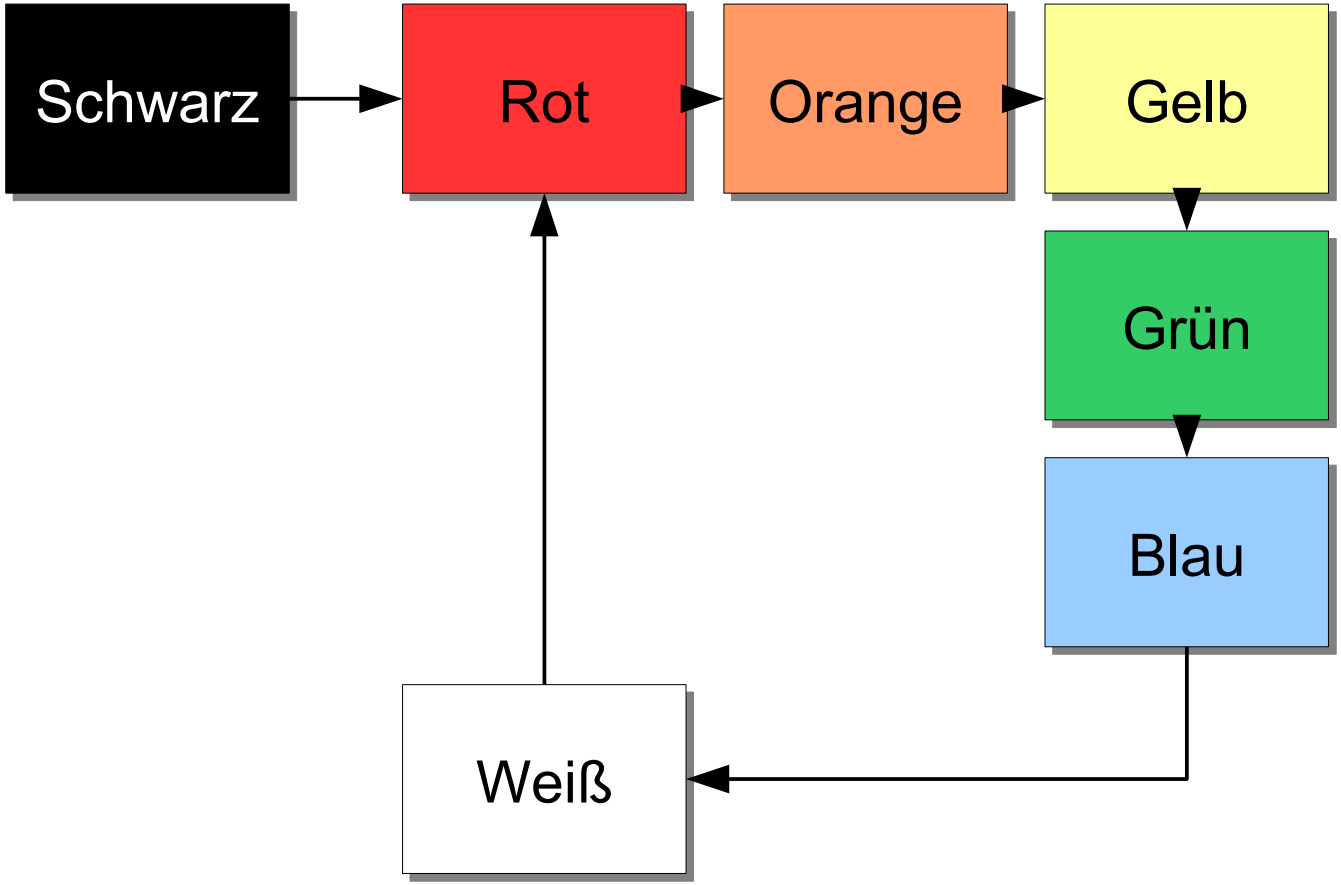
Das Wertesystem des CCD



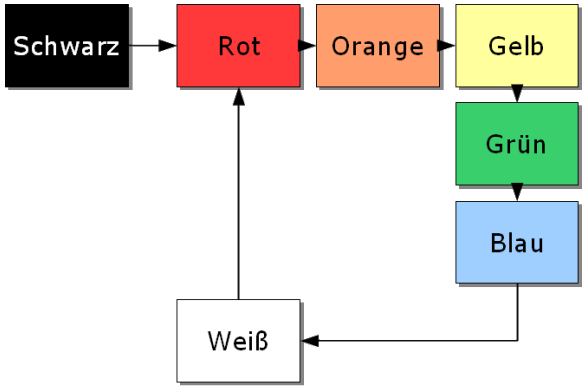
DHBW

Prinzipien und Praktiken des CCD

Aufteilen in Entwicklungsstufen



Das Grad-System des CCD



21 Tage Tragen eines Grades

Fokus auf dessen Inhalt

Tägliche Reflexion (Einschätzung)

Grad-Reihenfolge ohne Wertung

Nach weißem Grad nächster Zyklus

DHBW

Beispiel eines Prinzips

Keep it simple, stupid (KISS)

Einfache, direkte Lösungen

Unnötige Komplexität vermeiden

DHBW

Beispiel einer Praktik

Versionskontrollsystem einsetzen

Source Code ist immer gesichert

Änderungen mit minimalem Risiko

DHBW

Die Ziele des CCD

Wissen um die wichtigsten Themen

Üben und Bewußtwerden

Beständiges Verbessern

Verinnerlichen der Werte

Anwenden auch unter Stress

Vorteile für die Organisation

Langfristige Qualitätsverbesserung

Verantwortungsvolle Entwickler

Gemeinsames Fachverständnis

Attraktive Außenwirkung

Bessere Bewerberauswahl

DHBW

Vorteile für den Entwickler

Orientierungshilfe

Handlungssicherheit

Steigerung der Kompetenz

Effiziente Weiterbildung

DHBW


Clean Code Developer
Prinzipien und Praktiken für mehr Softwarequalität
www.clean-code-developer.de
Don´t Repeat Yourself (DRY)
Keep It Simple, Stupid (KISS)
Vorsicht vor Optimierungen
Favour Composition over Inheritance (FCoI)
Pfadfinderregel
Root Cause Analysis
Versionskontrolle
Einfache Refaktorisierungen
Täglich reflektieren
Entwurf und Impl. überlappen nicht
Implementation spiegelt Entwurf
You ain't gonna need it (YAGNI)
Continuous Integration (CI) II
Iterative Entwicklung
Komponentenorientierung
Test First
Single Level of Abstraction (SLA)
Single Responsibility Principle (SRP)
Separation of Concerns (SoC)
Sourcecode-Konventionen
Issue Tracking
Automatisierte Integrationstests
Lesen, Lesen, Lesen
Reviews
Open Closed Principle (OCP)
Tell don't ask
Law of Demeter (LoD)
Continuous Integration (CI) I
Statische Codeanalyse
Inversion of Control Container
Erfahrung weitergeben
Messen von Fehlern
Interface Segregation Principle (ISP)
Dependency Inversion Principle (DIP)
Liskov Substitution Principle (LSP)
Principle of Least Astonishment
Information Hiding Principle (IHP)
Automatisierte Unittests
Mockups
Code Coverage Analyse
Teilnahme an Fachveranstaltungen
Komplexe Refaktorisierungen
Evolvierbarkeit
Korrektheit
Produktionseffizienz
Reflexion
Single
Team
Prinzipien
Praktiken
08/2009
II PROFESSIONAL DEVELOPER COLLEGE
THE EXCITING WAY TO MORE .NET COMPETENCE
www.prodevcollege.de

DHBW

Details zu den Graden

Fünf Grade mit Fokusgruppen

Ein Grad (weiß) mit vollem Spektrum

Jeden Abend Reflektion:

„Habe ich die Prinzipien des aktuellen Grades eingehalten?“

Verinnerlichen eines Grades

21 Tage (~ein Arbeitsmonat) Fokus

Prinzipien nicht eingehalten? Zähler auf 0!

Gewöhnungseffekt nach 3 Wochen

Persönliche Ehrlichkeit, kein Wettkampf

DHBW

Fokus auf einen Grad ist Fortbildung

Fortbildung ist kein Urlaub

Fortbildung ist keine Nichtarbeit

Bildungsarbeit nutzbringend im Alltag

Empfehlung: 20% Bildungsarbeit

Fortbildung in der Softwareentwicklung

Vergleich anderer Berufsgruppen:

- Musiker üben und spielen nach
- Maler skizzieren und malen nach
- Chirurgen assistieren (üben an Modell)
- Piloten assistieren (üben im Simulator)
- Wissenschaftler lesen Fachliteratur

DHBW

Fortbildung in der Softwareentwicklung

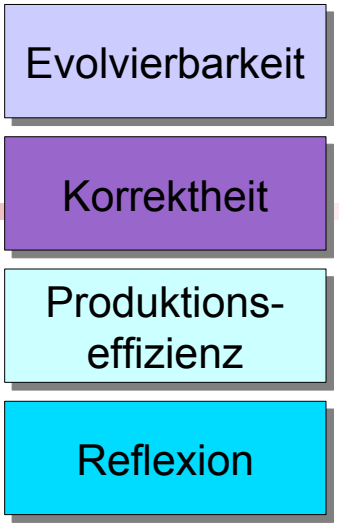
Das beste aus allen Welten wählen:

- Üben und nachprogrammieren
- Assistieren und gemeinsam entwickeln
- Lesen (gute Fachliteratur)

Gute Entwickler trainieren regelmäßig

DHBW

Die Werte des Clean Code Developers



Wertesystem als Leitfaden

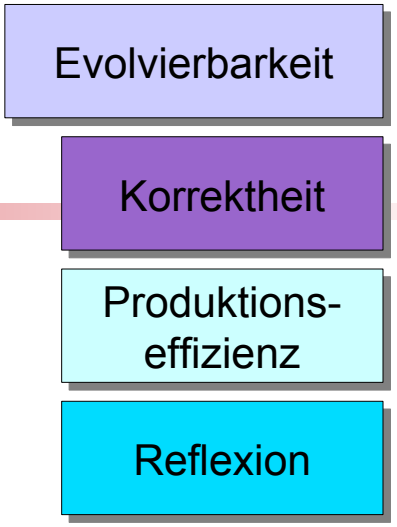
Rahmenbedingungen für Alltag

Prinzipien und Praktiken als Bausteine

Werte = Eigenanspruch aus Überzeugung

DHBW

Evolvierbarkeit als Wert



Sourcecode härtet wie Klebstoff

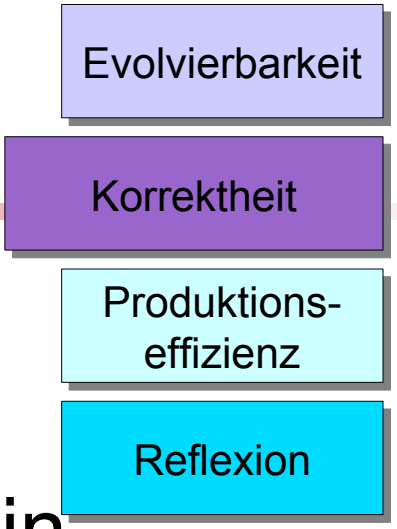
Ideal: Sourcecode wie Knetmasse

Stabil aber änderbar

Veränderbarkeit ist Grundanforderung

DHBW

Korrektheit als Wert



Anforderungen müssen erfüllt sein

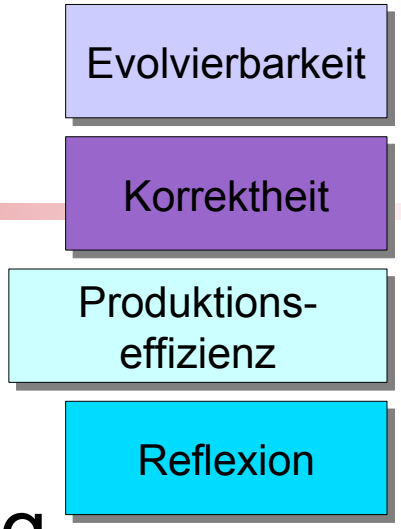
Anforderungen müssen definiert sein

Nachweis der Korrektheit ist Pflicht

Korrektheit als Entwicklertugend

DHBW

Produktionseffizienz als Wert



Ziel: minimale Kosten und Dauer

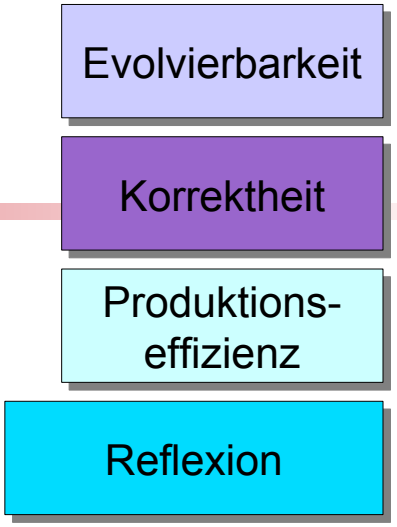
Zeitersparnis durch Automatisierung

Kostenersparnis durch niedrige Fehlerrate

Ökonomisches Maß für andere Werte

DHBW

Reflexion als Wert



Rückschau und Einsichtnahme

Lernen durch Evaluation

Informatik hält noch Erkenntnisse bereit

Ohne Reflexion keine Weiterbildung

DHBW

Die Grade des Clean Code Developers

Einfacher Einstieg, unverzichtbare Themen

Fundamentale Prinzipien, Automatisierung und Korrektheit

Automatisierte Unit-Tests, Objektorientierte Prinzipien

Codestrukturierung, Architektur, Kennzahlen

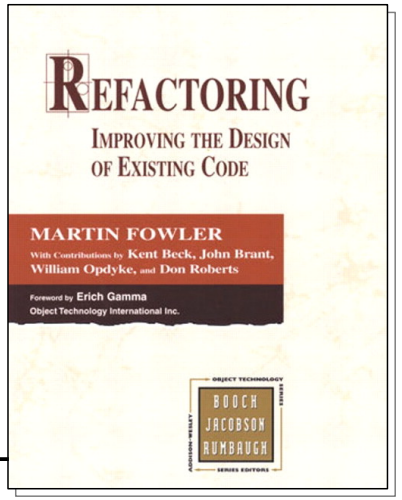
Vorgehensmodell, Deployment

DHBW

Der rote Grad im Detail

DHBW

Der rote Grad im Detail



DHBW

Der orange Grad im Detail

DHBW

Der orange Grad im Detail

I'm a Clean Coder!

PRINZIPIEN

- Single Level of Abstraction (SLA)
- Single Responsibility Principle (SRP)
- Separation of Concerns (SoC)
- Source Code Konventionen

PRAKTIKEN

- Issue Tracking
- Automatisierte Integrationstests
- Lesen, Lesen, Lesen
- Reviews

Single/Team · Evolvierbarkeit · Korrektheit · Produktionseffizienz · Reflexion

DHBW

Der gelbe Grad im Detail

DHBW

Der gelbe Grad im Detail

I'm a Clean Coder!

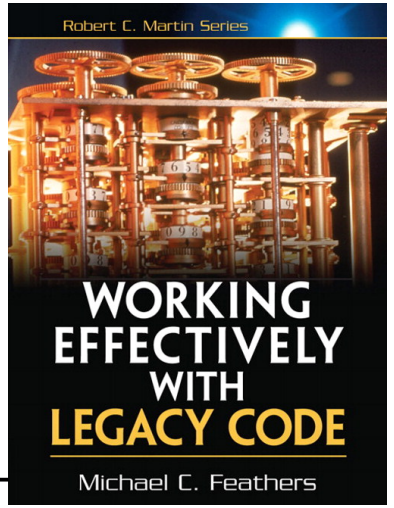
PRINZIPIEN

- Interface Segregation Principle (ISP)
- Dependency Inversion Principle
- Liskov Substitution Principle
- Principle of Least Astonishment
- Information Hiding Principle

PRAKTIKEN

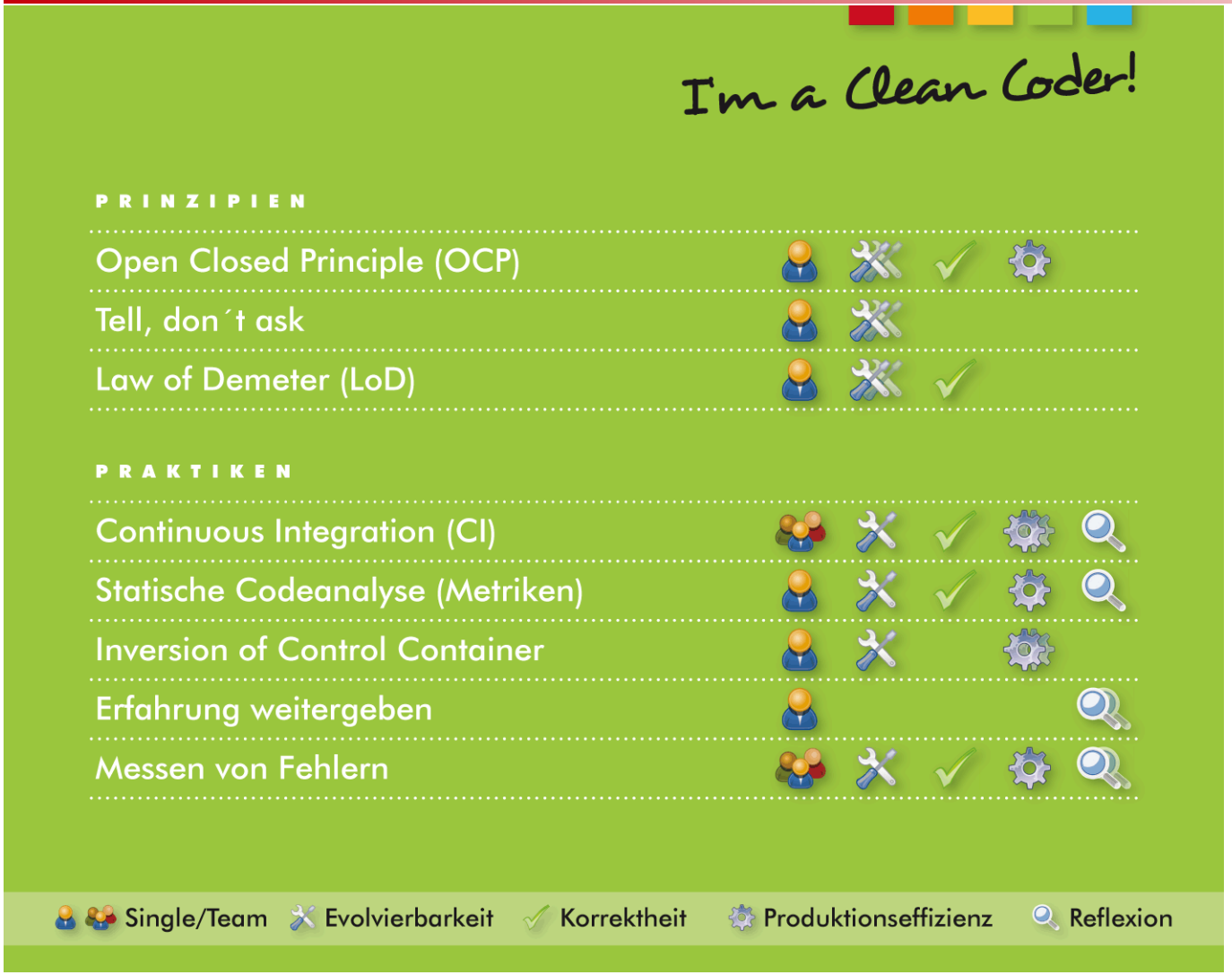
- Automatisierte Unit Tests
- Mockups (Testattrappen)
- Code Coverage Analyse
- Teilnahme an Fachveranstaltungen
- Komplexe Refaktorisierungen

Single/Team · Evolvierbarkeit · Korrektheit · Produktionseffizienz · Reflexion



DHBW

Der grüne Grad im Detail



DHBW

Der grüne Grad im Detail

I'm a Clean Coder!

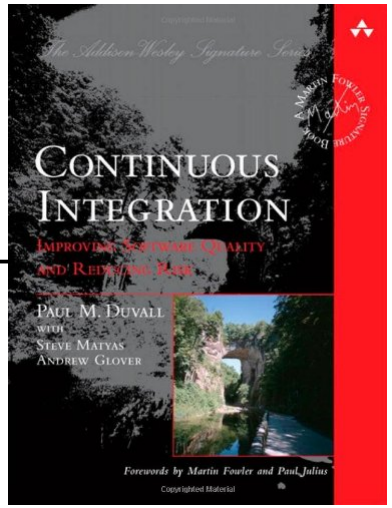
PRINZIPIEN

- Open Closed Principle (OCP)
- Tell, don´t ask
- Law of Demeter (LoD)

PRAKTIKEN

- Continuous Integration (CI)
- Statische Codeanalyse (Metriken)
- Inversion of Control Container
- Erfahrung weitergeben
- Messen von Fehlern

Single/Team · Evolvierbarkeit · Korrektheit · Produktionseffizienz · Reflexion



DHBW

Der blaue Grad im Detail

Der blaue Grad im Detail

I'm a Clean Coder!

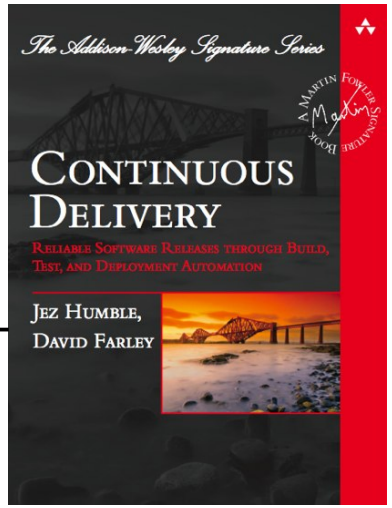
PRINZIPIEN

- Entwurf & Implementierung überlappen nicht
- Implementierung spiegelt Entwurf
- You Ain´t Gonna Need It (YAGNI)

PRAKTIKEN

- Continuous Delivery
- Iterative Entwicklung
- Komponentenorientierung
- Test First

Single/Team · Evolvierbarkeit · Korrektheit · Produktionseffizienz · Reflexion



DHBW

Der blaue Grad im Detail

I'm a Clean Coder!

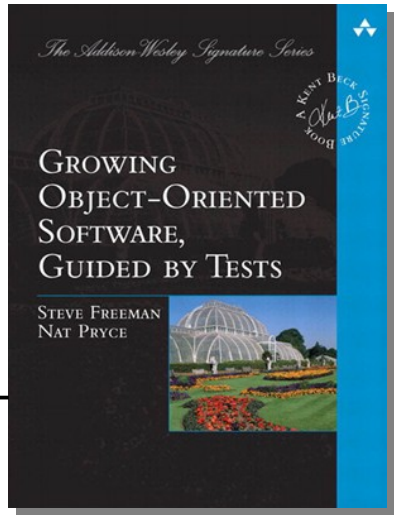
PRINZIPIEN

- Entwurf & Implementierung überlappen nicht
- Implementierung spiegelt Entwurf
- You Ain´t Gonna Need It (YAGNI)

PRAKTIKEN

- Continuous Delivery
- Iterative Entwicklung
- Komponentenorientierung
- Test First

Single/Team · Evolvierbarkeit · Korrektheit · Produktionseffizienz · Reflexion



DHBW

Der blaue Grad im Detail

I'm a Clean Coder!

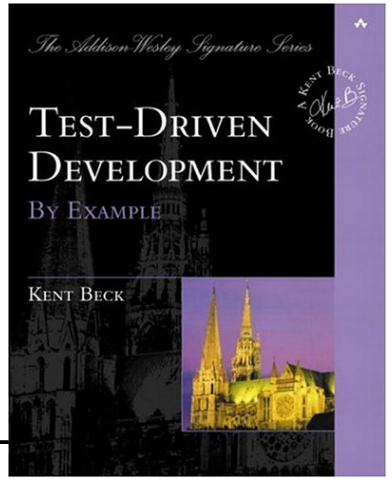
PRINZIPIEN

- Entwurf & Implementierung überlappen nicht
- Implementierung spiegelt Entwurf
- You Ain´t Gonna Need It (YAGNI)

PRAKTIKEN

- Continuous Delivery
- Iterative Entwicklung
- Komponentenorientierung
- Test First

Single/Team | Evolvierbarkeit | Korrektheit | Produktionseffizienz | Reflexion



DHBW

Der weiße Grad im Detail

I'm a Clean Coder!

PRINZIPIEN	PRAKTIKEN
• Don´t Repeat Yourself (DRY) • Keep it simple, stupid (KISS) • Vorsicht vor Optimierungen • Favour Composition over Inheritance (FCoI)	• Pfadfinderregel beachten • Root Cause Analysis • Versionskontrollsystem einsetzen • Einfache Refaktorisierungen • Täglich reflektieren
• Single Level of Abstraction (SLA) • Single Responsibility Principle (SRP) • Separation of Concerns (SoC) • Source Code Konventionen	• Issue Tracking • Automatisierte Integrationstests • Lesen, Lesen, Lesen • Reviews
• Interface Segregation Principle (ISP) • Dependency Inversion Principle • Liskov Substitution Principle • Principle of Least Astonishment • Information Hiding Principle	• Automatisierte Unit Tests • Mockups (Testattrappen) • Code Coverage Analyse • Teilnahme an Fachveranstaltungen • Komplexe Refaktorisierungen
• Open Closed Principle (OCP) • Tell, don´t ask • Law of Demeter (LoD)	• Continuous Integration (CI) • Statische Codeanalyse (Metriken) • Inversion of Control Container • Erfahrung weitergeben • Messen von Fehlern
• Entwurf und Implementierung überlappen nicht • Implementierung spiegelt Entwurf • You Ain´t Gonna Need It (YAGNI)	• Continuous Delivery • Iterative Entwicklung • Komponentenorientierung • Test First

http://clean-code-developer.de

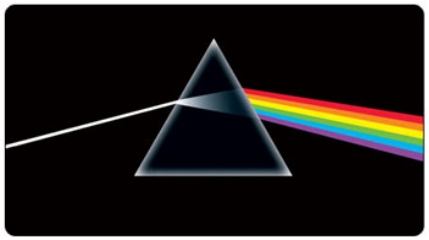
DHBW

Der weiße Grad im Detail

I'm a Clean Coder!

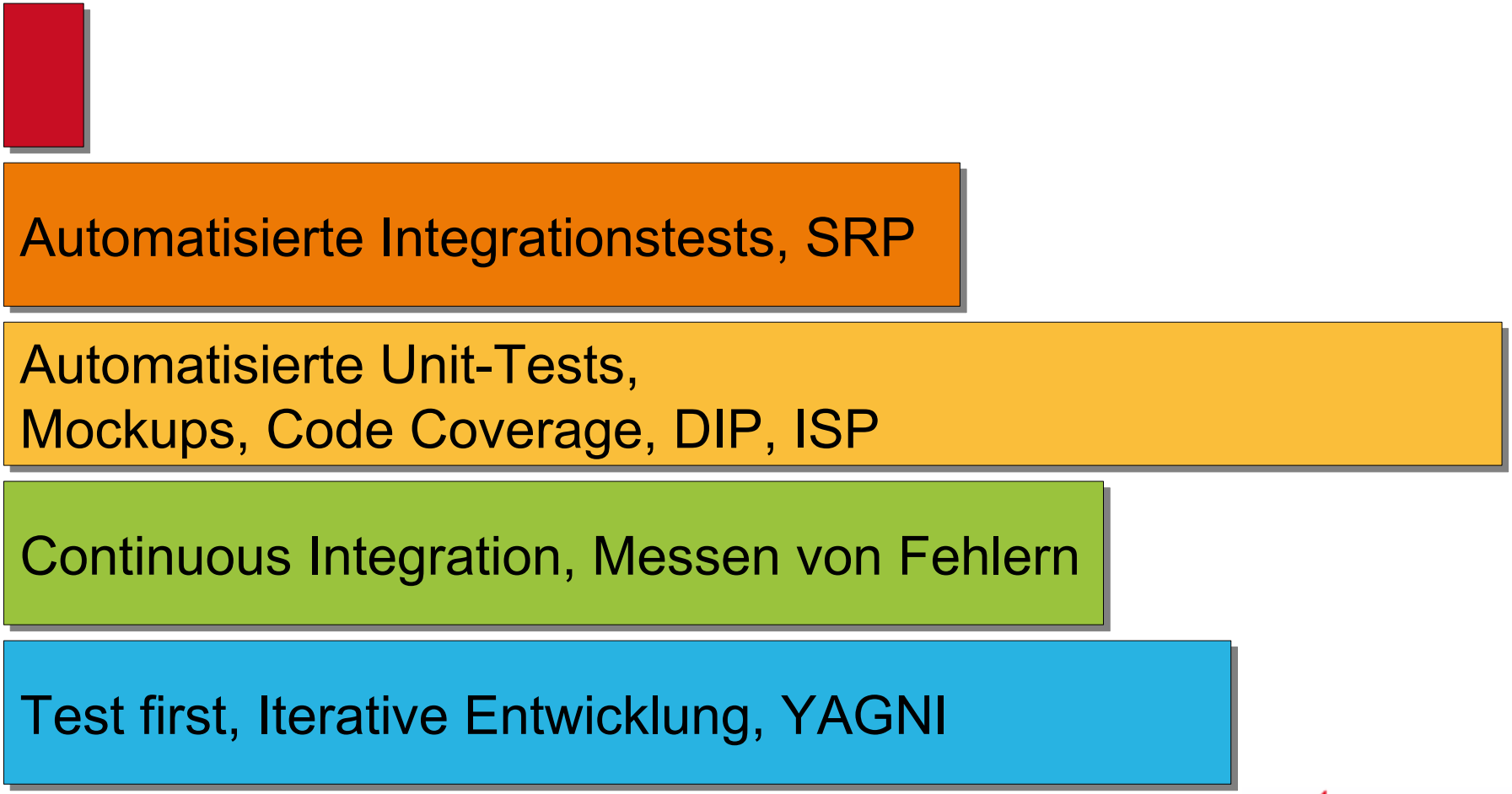
PRINZIPIEN	PRAKTIKEN
• Don´t Repeat Yourself (DRY) • Keep it simple, stupid (KISS) • Vorsicht vor Optimierungen • Favour Composition over Inheritance (FCoI)	• Pfadfinderregel beachten • Root Cause Analysis • Versionskontrollsystem einsetzen • Einfache Refaktorisierungen • Täglich reflektieren
• Single Level of Abstraction (SLA) • Single Responsibility Principle (SRP) • Separation of Concerns (SoC) • Source Code Konventionen	• Issue Tracking • Automatisierte Integrationstests • Lesen, Lesen, Lesen • Reviews
• Interface Segregation Principle (ISP) • Dependency Inversion Principle • Liskov Substitution Principle • Principle of Least Astonishment • Information Hiding Principle	• Automatisierte Unit Tests • Mockups (Testattrappen) • Code Coverage Analyse • Teilnahme an Fachveranstaltungen • Komplexe Refaktorisierungen
• Open Closed Principle (OCP) • Tell, don´t ask • Law of Demeter (LoD)	• Continuous Integration (CI) • Statische Codeanalyse (Metriken) • Inversion of Control Container • Erfahrung weitergeben • Messen von Fehlern
• Entwurf und Implementierung überlappen nicht • Implementierung spiegelt Entwurf • You Ain´t Gonna Need It (YAGNI)	• Continuous Delivery • Iterative Entwicklung • Komponentenorientierung • Test First

http://clean-code-developer.de



DHBW

Lernpfade durch die Grade: Korrektheit



DHBW

Lernpfade durch die Grade: Evolvierbarkeit

Einfache Refaktorisierungen, DRY, KISS

Issue Tracking, Source Code Konventionen, SoC

Komplexe Refaktorisierungen, Information Hiding Principle

Metriken, Law of Demeter, Tell, don't ask

Komponentenorientierung

DHBW

Internet-Ressourcen

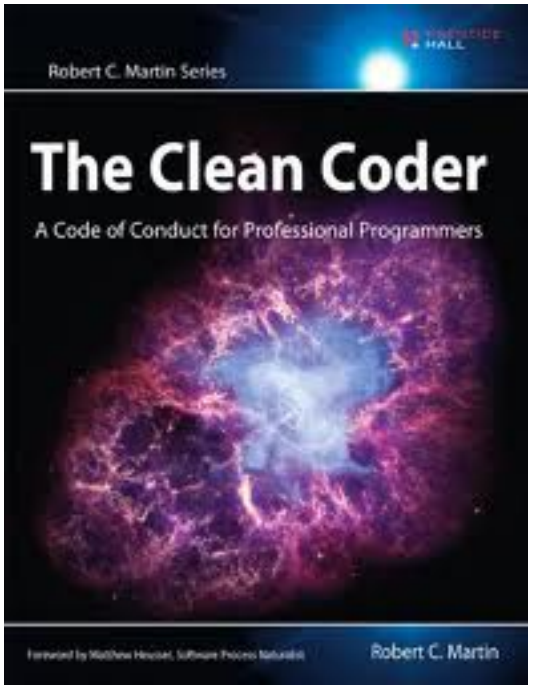
http://www.clean-code-developer.de

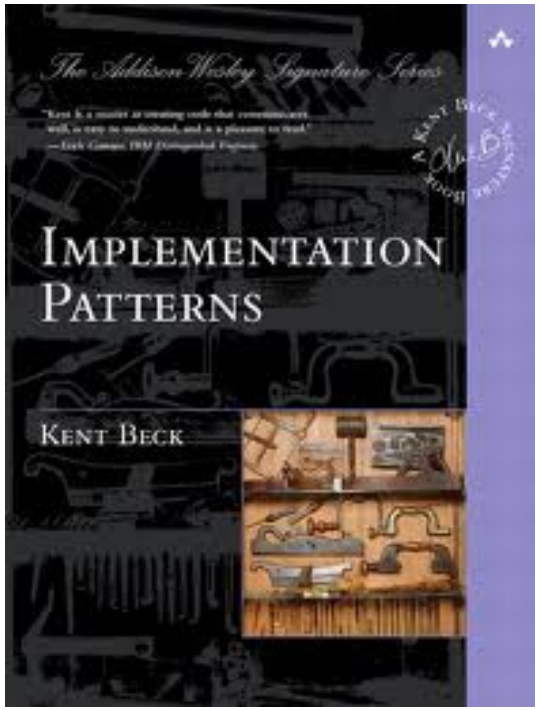
Community-Seite, Fokus auf .NET

Diskussionsgruppen auf Google und XING

DHBW

Literatur direkt zum Thema





Das Reverse-Prisma