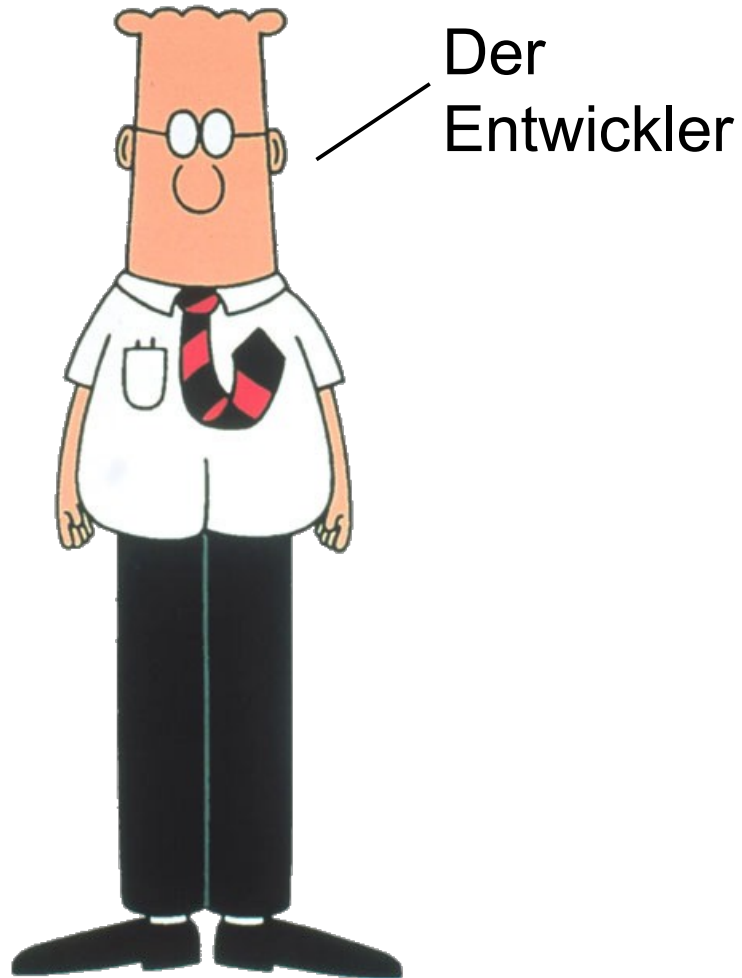


Die `clean code developer` Initiative (CCD)

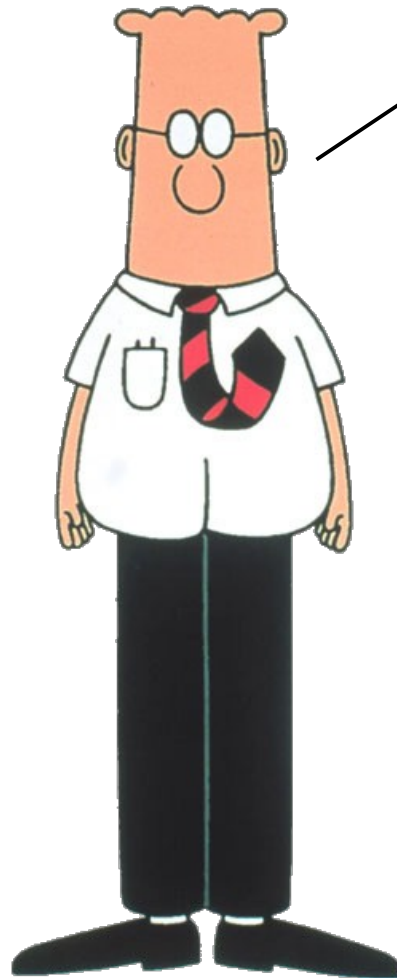
Softwareentwicklung

Softwareentwicklung wird von Menschen ausgeführt

Softwareentwicklung



Softwareentwicklung



Der professionelle
Entwickler

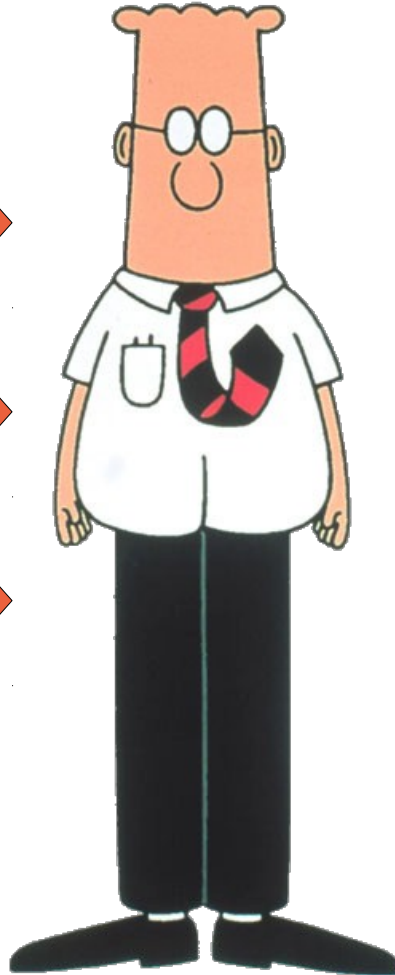
Softwareentwicklung

Äußere Einflüsse

Termindruck

Budgetzwang

Qualitäts-
anforderung



Projektdreieck



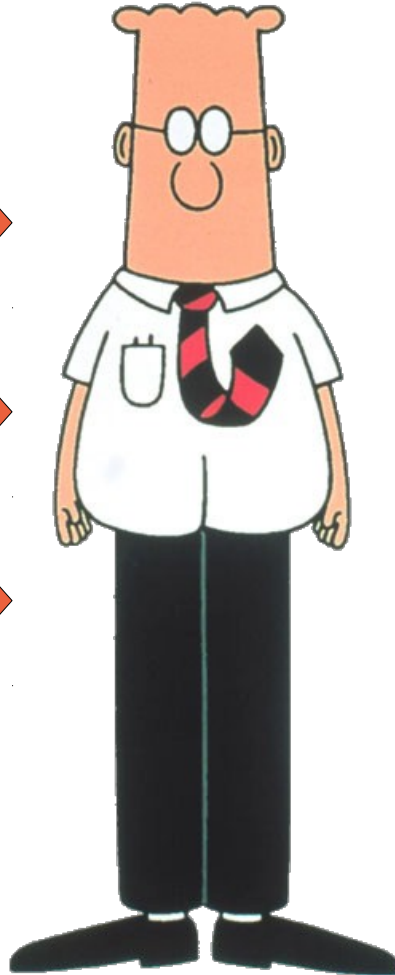
Softwareentwicklung

Äußere Einflüsse

Termindruck

Budgetzwang

Qualitäts-
anforderung



Innere Einflüsse

Wissen &
Erfahrung

Prozess

Werte

Das Wertesystem des CCD

Evolvierbarkeit

Korrektheit

Produktions-
effizienz

Reflexion

Prinzipien

Praktiken

Prinzipien und Praktiken des CCD

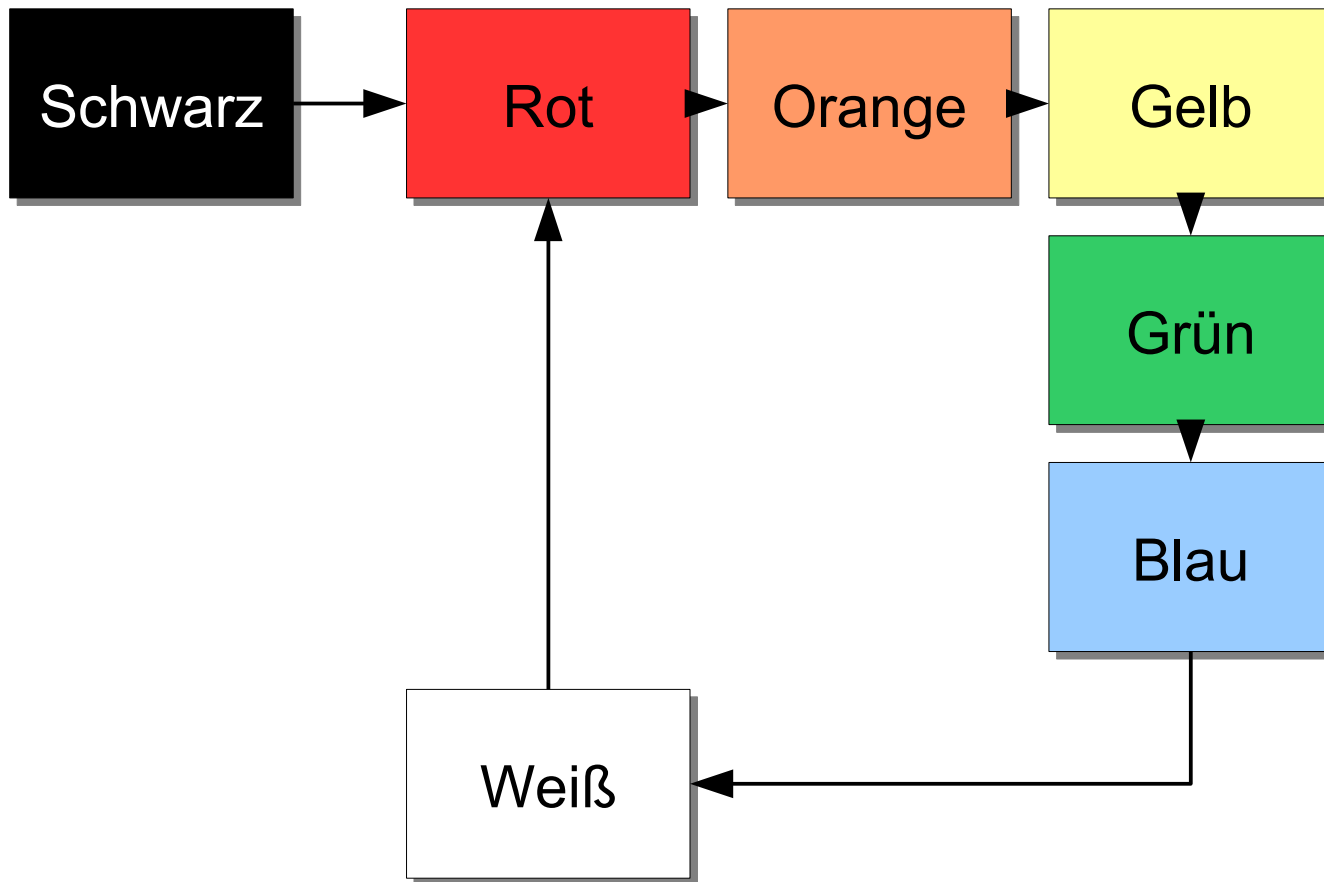
19 Prinzipien

Entwurf und Implementierung überlappen nicht
Keep it simple, stupid (KISS)
Don't Repeat Yourself (DRY)
Implement the interface, not the abstraction
Single Level of Abstraction (SLA)
Interface Segregation Principle (ISP)
You Ain't Gonna Have Them All
Principle of Inheritance (FCoI)
Dependency Inversion Principle
Single Responsibility Principle
Tell don't ask
Vorsicht vor Optimierungen!

23 Praktiken

Reviews
Issue Tracking
Teilnahme an Fachkonferenzen
Ein Versionskontrollsystem einsetzen
Mockups
Einfache, effektive Tests
Erfahrung weitergeben
Täglich reflektieren
Statische Codeanalyse
Continuous Delivery
Root Tests
Komponentenkomposition
Automatisierte Regressionstests
Die Pfadfinderregel beachten
Lesen, Lesen, Lesen
Iterative Entwicklung
Codeanalyse (Metriken)
Continuous Delivery
Entwicklungsprozess
Komplexitätsanalyse
Refactoring

Aufteilen in Entwicklungsstufen



Das Grad-System des CCD

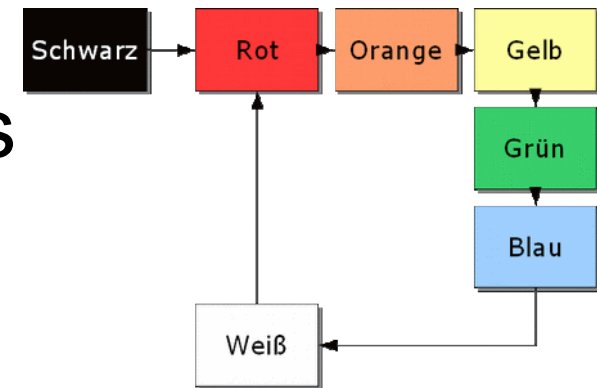
21 Tage Tragen eines Grades

Fokus auf dessen Inhalt

Tägliche Reflexion (Einschätzung)

Grad-Reihenfolge ohne Wertung

Nach weißem Grad nächster Zyklus



Beispiel eines Prinzips

Keep it simple, stupid
(KISS)



I'm a Clean Coder!

PRINZIPIEN					
Don't Repeat Yourself (DRY)	Person	Checkmark	Checkmark	Gear	Magnifying Glass
Keep it simple, stupid (KISS)	Person	Checkmark	Checkmark	Gear	Magnifying Glass
Vorsicht vor Optimierungen	Person	Checkmark	Checkmark	Gear	Magnifying Glass
Favour Composition over Inheritance (FCoI)	Person	Checkmark	Checkmark	Gear	Magnifying Glass
PRAKTIKEN					
Pfadfinderregel beachten	Person	Checkmark	Checkmark	Gear	Magnifying Glass
Root Cause Analysis	Person	Checkmark	Checkmark	Gear	Magnifying Glass
Versionskontrollsystem einsetzen	Person	Checkmark	Checkmark	Gear	Magnifying Glass
Einfache Refaktorisierungen	Person	Checkmark	Checkmark	Gear	Magnifying Glass
Täglich reflektieren	Person	Checkmark	Checkmark	Gear	Magnifying Glass

Single/Team Evolvierbarkeit Korrektheit Produktionseffizienz Reflexion

Einfache, direkte Lösungen

Unnötige Komplexität vermeiden

Beispiel einer Praktik

Versionskontrollsystem einsetzen



Source Code ist immer gesichert
Änderungen mit minimalem Risiko

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

Vorteile für den Entwickler

Orientierungshilfe

Handlungssicherheit

Steigerung der Kompetenz

Effiziente Weiterbildung



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

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

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

Fortbildung in der Softwareentwicklung

Das beste aus allen Welten wählen:

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

Gute Entwickler trainieren regelmäßig

Die Werte des Clean Code Developers

Evolvierbarkeit

Korrektheit

Produktions-
effizienz

Reflexion

Wertesystem als Leitfaden

Rahmenbedingungen für Alltag

Prinzipien und Praktiken als Bausteine

Werte = Eigenanspruch aus Überzeugung

Evolvierbarkeit als Wert

Evolvierbarkeit

Korrektheit

Produktions-
effizienz

Reflexion

Sourcecode härtet wie Klebstoff

Ideal: Sourcecode wie Knetmasse

Stabil aber änderbar

Veränderbarkeit ist Grundanforderung

Korrektheit als Wert

Evolvierbarkeit

Korrektheit

Produktions-
effizienz

Reflexion

Anforderungen müssen erfüllt sein

Anforderungen müssen definiert sein

Nachweis der Korrektheit ist Pflicht

Korrektheit als Entwicklertugend

Produktionseffizienz als Wert

Evolvierbarkeit

Korrektheit

Produktions-
effizienz

Reflexion

Ziel: minimale Kosten und Dauer

Zeitersparnis durch Automatisierung

Kostensparnis durch niedrige Fehlerrate

Ökonomisches Maß für andere Werte

Reflexion als Wert

Evolvierbarkeit

Korrektheit

Produktions-
effizienz

Reflexion

Rückschau und Einsichtnahme

Lernen durch Evaluation

Informatik hält noch Erkenntnisse bereit

Ohne Reflexion keine Weiterbildung

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

Der rote Grad im Detail

I'm a Clean Coder!

PRINZIPIEN

Don't Repeat Yourself (DRY)



Keep it simple, stupid (KISS)



Vorsicht vor Optimierungen



Favour Composition over Inheritance (FCoI)



PRAKTIKEN

Pfadfinderregel beachten



Root Cause Analysis



Versionskontrollsystem einsetzen



Einfache Refaktorisierungen



Täglich reflektieren



Der rote Grad im Detail

I'm a Clean Coder!

PRINZIPIEN

Don't Repeat Yourself (DRY)



Keep it simple, stupid (KISS)



Vorsicht vor Optimierungen



Favour Composition over Inheritance (FCoI)



PRAKTIKEN

Pfadfinderregel beachten



Root Cause Analysis



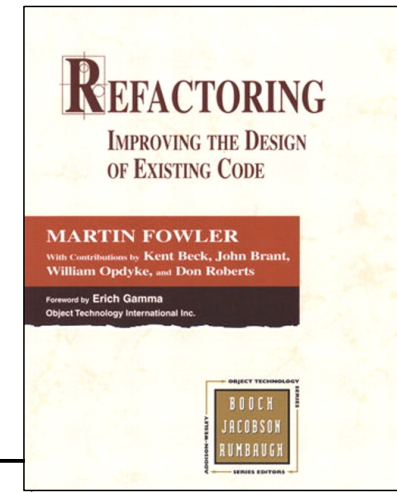
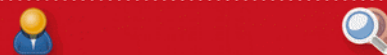
Versionskontrollsystem einsetzen



Einfache Refaktorisierungen



Täglich reflektieren



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



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



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 (Testattractanten)



Code Coverage Analyse



Teilnahme an Fachveranstaltungen



Komplexe Refaktorisierungen



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)



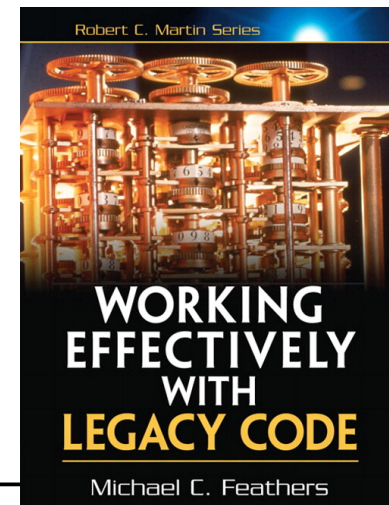
Code Coverage Analyse



Teilnahme an Fachveranstaltungen



Komplexe Refaktorisierungen



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



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)



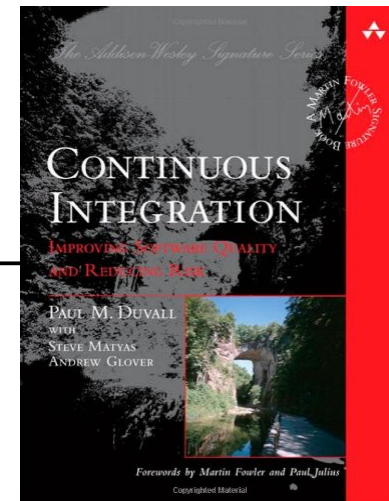
Inversion of Control Container



Erfahrung weitergeben



Messen von Fehlern



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				 




















Der blaue Grad im Detail

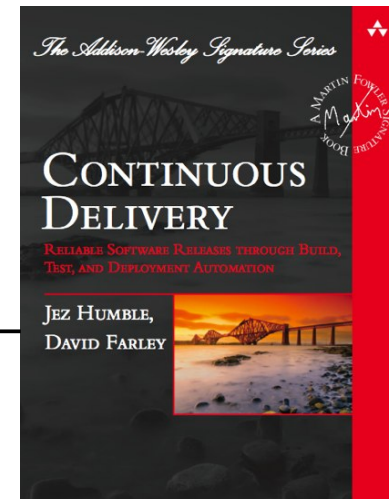
I'm a Clean Coder!

PRINZIPIEN

Entwurf & Implementierung überlappen nicht				
Implementierung spiegelt Entwurf				
You Ain't Gonna Need It (YAGNI)				 

PRAKTIKEN

Continuous Delivery				 
Iterative Entwicklung				 
Komponentenorientierung				
Test First				 



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



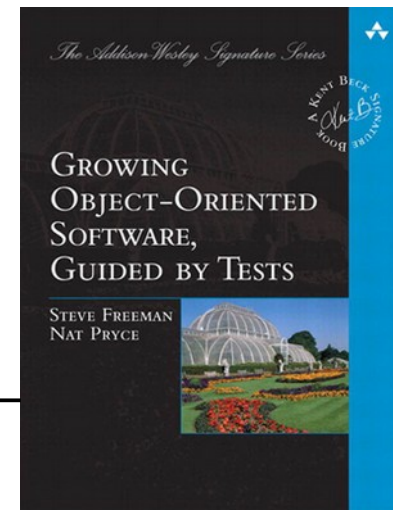
Iterative Entwicklung



Komponentenorientierung



Test First



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



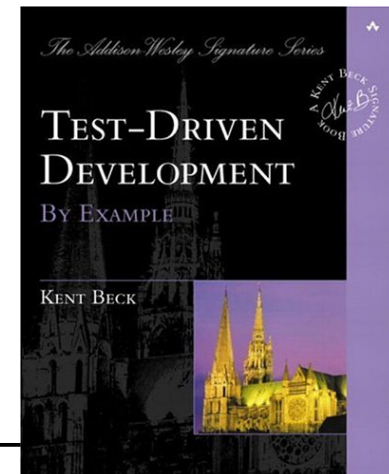
Iterative Entwicklung



Komponentenorientierung



Test First



Der weiße Grad im Detail

I'm a Clean Coder!

PRINZIPIEN

PRAKTIKEN

<ul style="list-style-type: none"> ● Don't Repeat Yourself (DRY) ● Keep it simple, stupid (KISS) ● Vorsicht vor Optimierungen ● Favour Composition over Inheritance (FCoI) 	<ul style="list-style-type: none"> ● Pfadfinderregel beachten ● Root Cause Analysis ● Versionskontrollsystem einsetzen ● Einfache Refaktorisierungen ● Täglich reflektieren 	
<ul style="list-style-type: none"> ● Single Level of Abstraction (SLA) ● Single Responsibility Principle (SRP) ● Separation of Concerns (SoC) ● Source Code Konventionen 	<ul style="list-style-type: none"> ● Issue Tracking ● Automatisierte Integrationstests ● Lesen, Lesen, Lesen ● Reviews 	
<ul style="list-style-type: none"> ● Interface Segregation Principle (ISP) ● Dependency Inversion Principle ● Liskov Substitution Principle ● Principle of Least Astonishment ● Information Hiding Principle 	<ul style="list-style-type: none"> ● Automatisierte Unit Tests ● Mockups (Testattrappen) ● Code Coverage Analyse ● Teilnahme an Fachveranstaltungen ● Komplexe Refaktorisierungen 	
<ul style="list-style-type: none"> ● Open Closed Principle (OCP) ● Tell, don't ask ● Law of Demeter (LoD) 	<ul style="list-style-type: none"> ● Continuous Integration (CI) ● Statische Codeanalyse (Metriken) ● Inversion of Control Container ● Erfahrung weitergeben ● Messen von Fehlern 	
<ul style="list-style-type: none"> ● Entwurf und Implementierung überlappen nicht ● Implementierung spiegelt Entwurf ● You Ain't Gonna Need It (YAGNI) 	<ul style="list-style-type: none"> ● Continuous Delivery ● Iterative Entwicklung ● Komponentenorientierung ● Test First 	

Der weiße Grad im Detail

I'm a Clean Coder!

PRINZIPIEN

PRAKTIKEN

<ul style="list-style-type: none"> ● Don't Repeat Yourself (DRY) ● Keep it simple, stupid (KISS) ● Vorsicht vor Optimierungen ● Favour Composition over Inheritance (FCoI) 	<ul style="list-style-type: none"> ● Pfadfinderregel beachten ● Root Cause Analysis ● Versionskontrollsystem einsetzen ● Einfache Refaktorisierungen ● Täglich reflektieren 	
<ul style="list-style-type: none"> ● Single Level of Abstraction (SLA) ● Single Responsibility Principle (SRP) ● Separation of Concerns (SoC) ● Source Code Konventionen 	<ul style="list-style-type: none"> ● Issue Tracking ● Automatisierte Integrationstests ● Lesen, Lesen, Lesen ● Reviews 	
<ul style="list-style-type: none"> ● Interface Segregation Principle (ISP) ● Dependency Inversion Principle ● Liskov Substitution Principle ● Principle of Least Astonishment ● Information Hiding Principle 	<ul style="list-style-type: none"> ● Automatisierte Unit Tests ● Mockups (Testattrappen) ● Code Coverage Analyse ● Teilnahme an Fachveranstaltungen ● Komplexe Refaktorisierungen 	
<ul style="list-style-type: none"> ● Open Closed Principle (OCP) ● Tell, don't ask ● Law of Demeter (LoD) 	<ul style="list-style-type: none"> ● Continuous Integration (CI) ● Statische Codeanalyse (Metriken) ● Inversion of Control Container ● Erfahrung weitergeben ● Messen von Fehlern 	
<ul style="list-style-type: none"> ● Entwurf und Implementierung überlappen nicht ● Implementierung spiegelt Entwurf ● You Ain't Gonna Need It (YAGNI) 	<ul style="list-style-type: none"> ● Continuous Delivery ● Iterative Entwicklung ● Komponentenorientierung ● Test First 	



Lernpfade durch die Grade: Korrektheit



Automatisierte Integrationstests, SRP

Automatisierte Unit-Tests,
Mockups, Code Coverage, DIP, ISP

Continuous Integration, Messen von Fehlern

Test first, Iterative Entwicklung, YAGNI

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

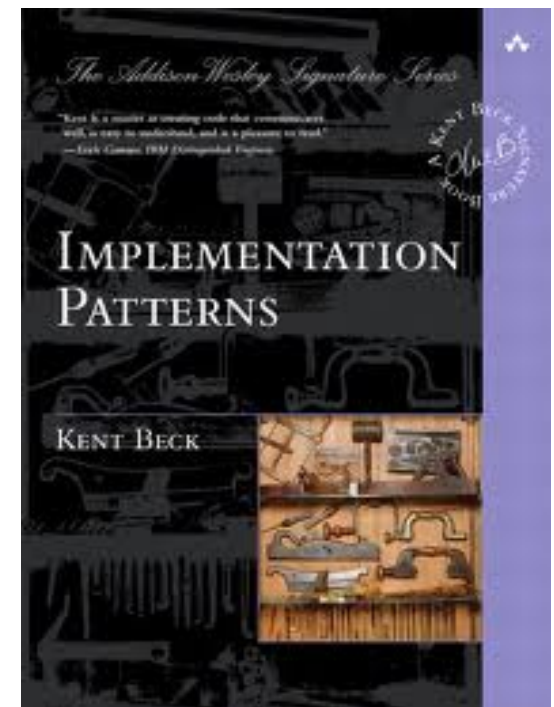
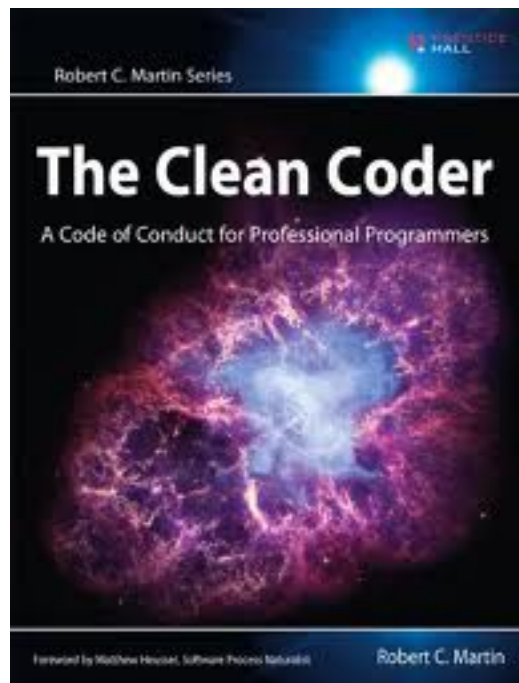
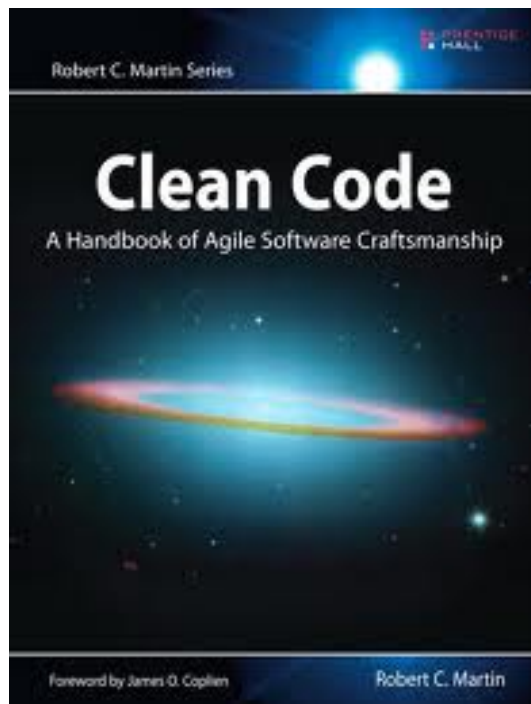
Internet-Ressourcen

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

Community-Seite, Fokus auf .NET

Diskussionsgruppen auf Google und XING

Literatur direkt zum Thema



Das Reverse-Prisma

