

**Die**  
**clean code**  
**developer**  
**Initiative**  
**(CCD)**

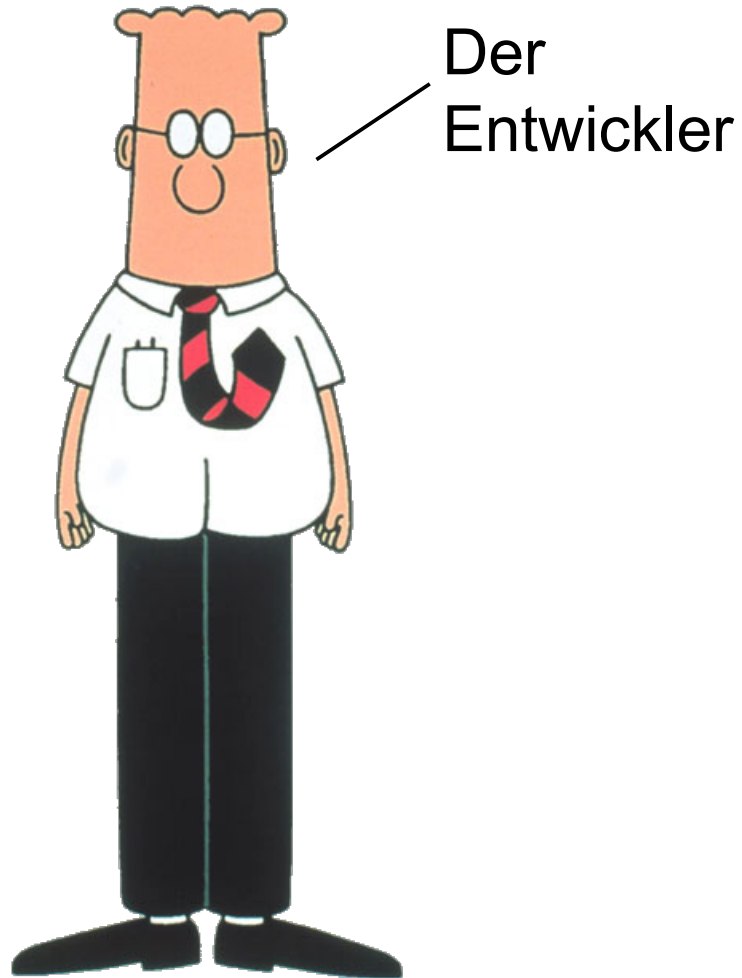
# Softwareentwicklung

---

Softwareentwicklung wird von Menschen ausgeführt

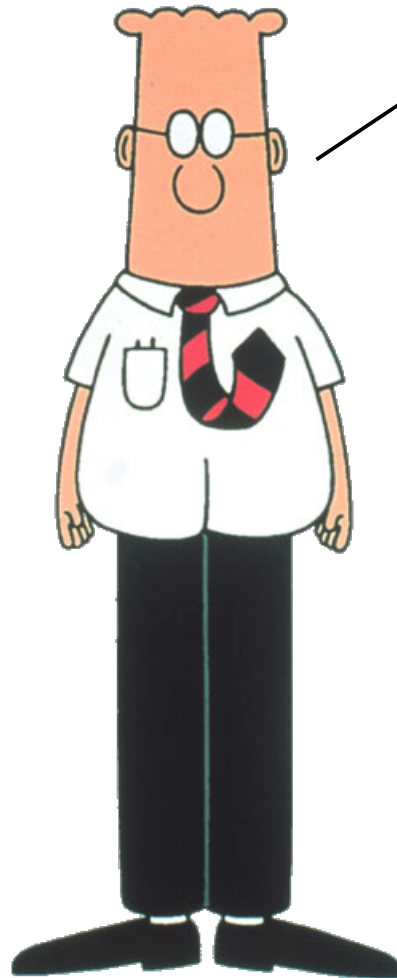
# Softwareentwicklung

---



# Softwareentwicklung

---



Der professionelle  
Entwickler

# Softwareentwicklung

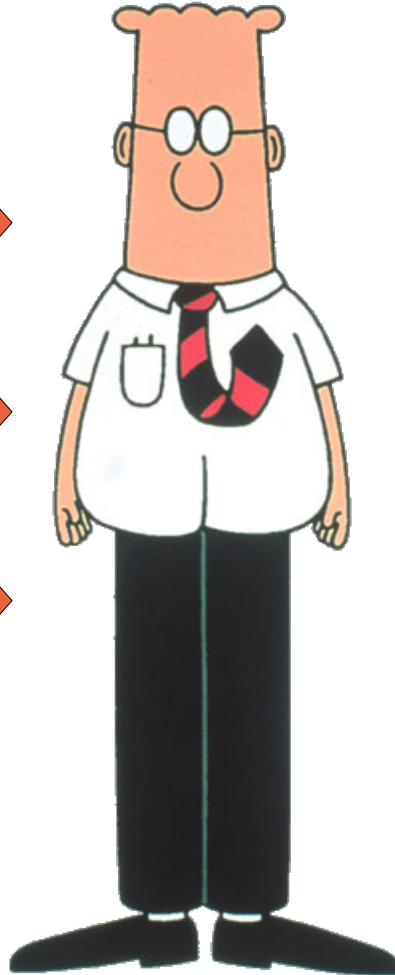
---

Äußere Einflüsse

Termindruck

Budgetzwang

Qualitäts-  
anforderung



# Projektdreieck

---



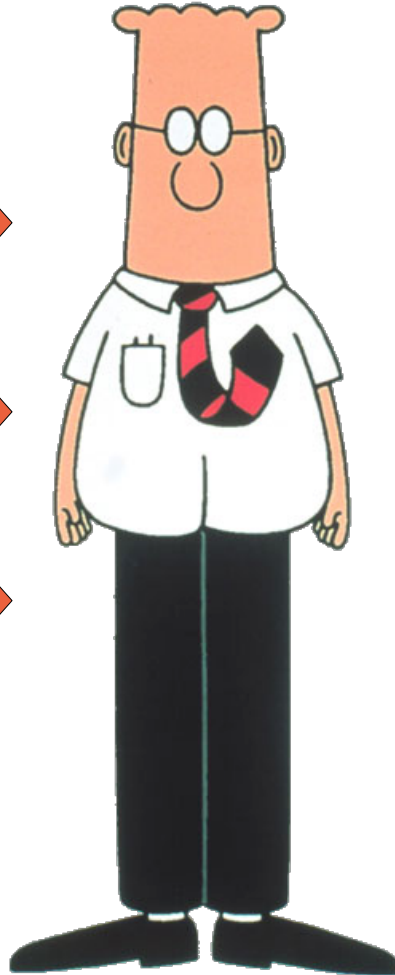
# Softwareentwicklung

## 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

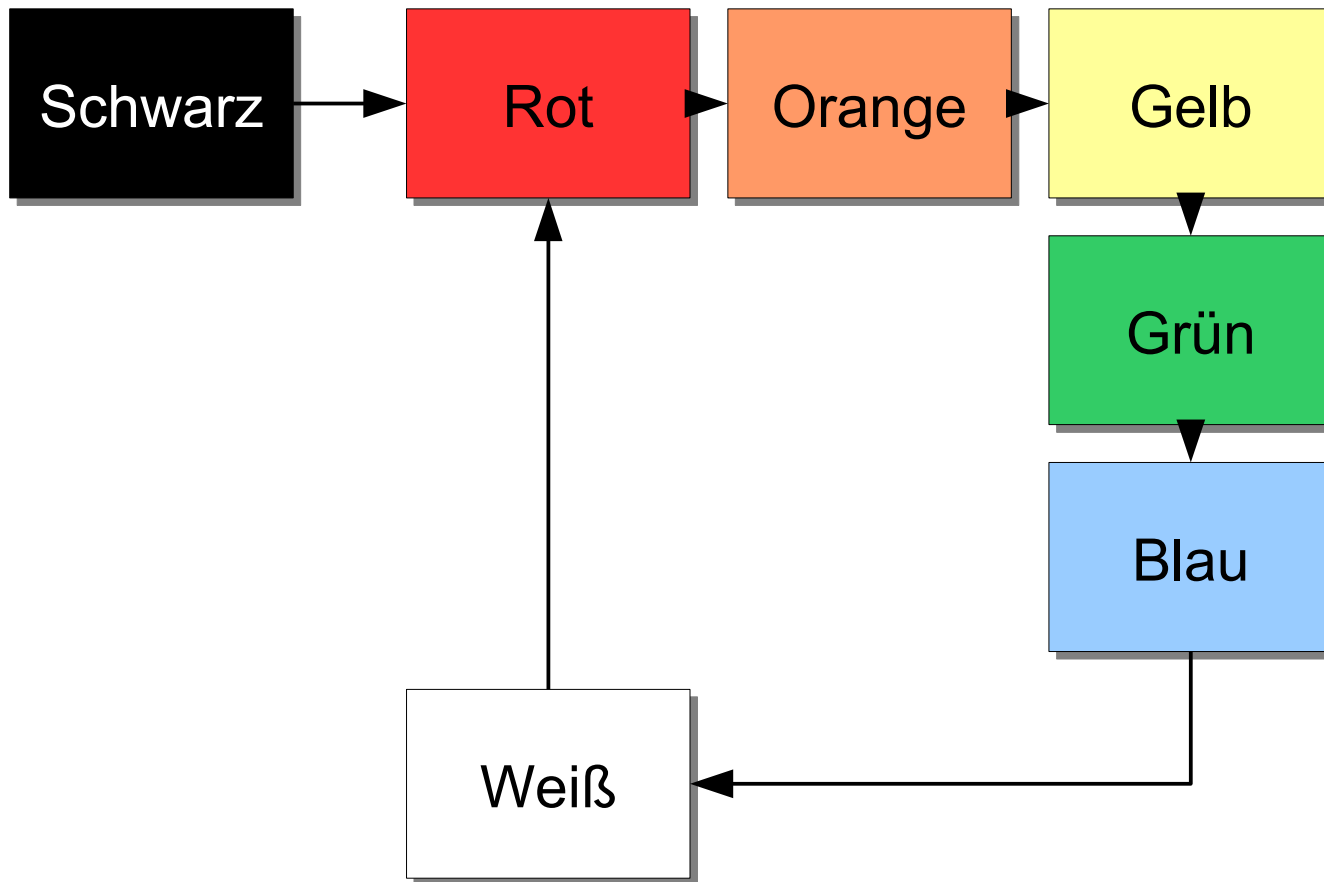
## 19 Prinzipien

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 Overlap  
Liskov Substitution (FCoI)  
Dependency Inversion Principle  
Single Responsibility Principle (SRP)  
Vorsicht vor Optimierungen!

## 23 Praktiken

Reviews  
Issue Tracking  
Teilnahme an Fachveranstaltungen  
Ein Versionskontrollsystem einsetzen  
Mockups  
Einfache, effektive Code Coverage Analyse  
Erfahrung weitergeben  
Komponententests  
Täglich reflektieren  
Automatisierte Regressionstests  
Statische Codeanalyse (Metriken)  
Continuous Delivery  
Entwicklungszyklen  
Root Cause Analysis  
Die Pfadfinderregel beachten  
Lesen, Lesen, Lesen

# Aufteilen in Entwicklungsstufen



# Das Grad-System des CCD

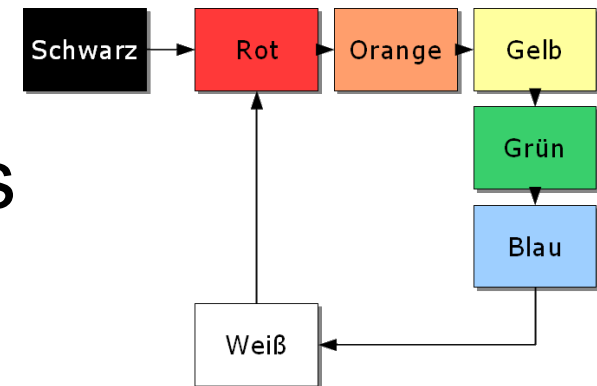
21 Tage Tragen eines Grades

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



*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                       |  |  |  |  |

Single/Team Evolvierbarkeit Korrektheit Produktionseffizienz Reflexion

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)                | 1 | 2 | 3 | 4 | 5 |
| Keep It Simple, Stupid (KISS)              | 1 | 2 | 3 | 4 | 5 |
| Vorsicht vor Optimierungen                 | 1 | 2 | 3 | 4 | 5 |
| Favour Composition over Inheritance (FCoI) | 1 | 2 | 3 | 4 | 5 |
| Pfadfinderregel                            | 1 | 2 | 3 | 4 | 5 |
| Root Cause Analysis                        | 1 | 2 | 3 | 4 | 5 |
| Versionskontrolle                          | 1 | 2 | 3 | 4 | 5 |
| Einfache Refaktorisierungen                | 1 | 2 | 3 | 4 | 5 |
| Täglich reflektieren                       | 1 | 2 | 3 | 4 | 5 |

|                                       |   |   |   |   |   |
|---------------------------------------|---|---|---|---|---|
| Single Level of Abstraction (SLA)     | 1 | 2 | 3 | 4 | 5 |
| Single Responsibility Principle (SRP) | 1 | 2 | 3 | 4 | 5 |
| Separation of Concerns (SoC)          | 1 | 2 | 3 | 4 | 5 |
| Sourcecode-Konventionen               | 1 | 2 | 3 | 4 | 5 |
| Issue Tracking                        | 1 | 2 | 3 | 4 | 5 |
| Automatisierte Integrationstests      | 1 | 2 | 3 | 4 | 5 |
| Lesen, Lesen, Lesen                   | 1 | 2 | 3 | 4 | 5 |
| Reviews                               | 1 | 2 | 3 | 4 | 5 |

|                                       |   |   |   |   |   |
|---------------------------------------|---|---|---|---|---|
| Interface Segregation Principle (ISP) | 1 | 2 | 3 | 4 | 5 |
| Dependency Inversion Principle (DIP)  | 1 | 2 | 3 | 4 | 5 |
| Liskov Substitution Principle (LSP)   | 1 | 2 | 3 | 4 | 5 |
| Principle of Least Astonishment       | 1 | 2 | 3 | 4 | 5 |
| Information Hiding Principle (IHP)    | 1 | 2 | 3 | 4 | 5 |
| Automatisierte Unittests              | 1 | 2 | 3 | 4 | 5 |
| Mockups                               | 1 | 2 | 3 | 4 | 5 |
| Code Coverage Analyse                 | 1 | 2 | 3 | 4 | 5 |
| Teilnahme an Fachveranstaltungen      | 1 | 2 | 3 | 4 | 5 |
| Komplexe Refaktorisierungen           | 1 | 2 | 3 | 4 | 5 |

|                                    |   |   |   |   |   |
|------------------------------------|---|---|---|---|---|
| Entwurf und Impl. überlappen nicht | 1 | 2 | 3 | 4 | 5 |
| Implementation spiegelt Entwurf    | 1 | 2 | 3 | 4 | 5 |
| You ain't gonna need it (YAGNI)    | 1 | 2 | 3 | 4 | 5 |
| Continuous Integration (CI) II     | 1 | 2 | 3 | 4 | 5 |
| Iterative Entwicklung              | 1 | 2 | 3 | 4 | 5 |
| Komponentenorientierung            | 1 | 2 | 3 | 4 | 5 |
| Test First                         | 1 | 2 | 3 | 4 | 5 |

|                                |   |   |   |   |   |
|--------------------------------|---|---|---|---|---|
| Open Closed Principle (OCP)    | 1 | 2 | 3 | 4 | 5 |
| Tell don't ask                 | 1 | 2 | 3 | 4 | 5 |
| Law of Demeter (LoD)           | 1 | 2 | 3 | 4 | 5 |
| Continuous Integration (CI) I  | 1 | 2 | 3 | 4 | 5 |
| Statische Codeanalyse          | 1 | 2 | 3 | 4 | 5 |
| Inversion of Control Container | 1 | 2 | 3 | 4 | 5 |
| Erfahrung weitergeben          | 1 | 2 | 3 | 4 | 5 |
| Messen von Fehlern             | 1 | 2 | 3 | 4 | 5 |

|                      |   |   |   |   |   |
|----------------------|---|---|---|---|---|
| Evolvierbarkeit      | 1 | 2 | 3 | 4 | 5 |
| Korrektheit          | 1 | 2 | 3 | 4 | 5 |
| Produktionseffizienz | 1 | 2 | 3 | 4 | 5 |
| Reflexion            | 1 | 2 | 3 | 4 | 5 |
| Single               | 1 | 2 | 3 | 4 | 5 |
| Team                 | 1 | 2 | 3 | 4 | 5 |
| Prinzipien           | 1 | 2 | 3 | 4 | 5 |
| Praktiken            | 1 | 2 | 3 | 4 | 5 |

## 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



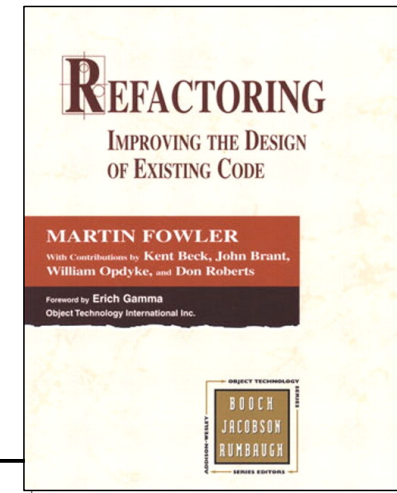
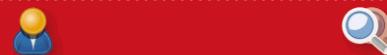
Versionskontrollsystem einsetzen



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 (Testatruppen)



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)



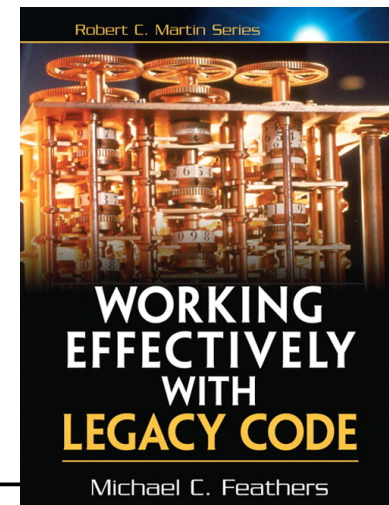
Code Coverage Analyse



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)



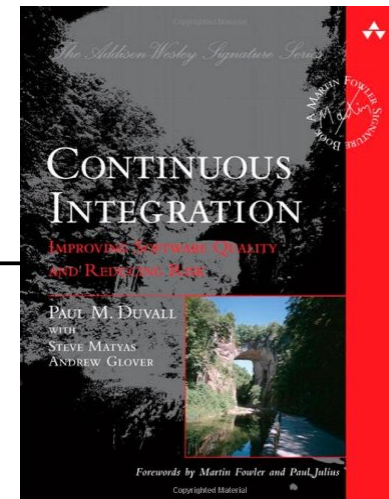
Inversion of Control Container



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)            |  |  |  | <br> |

## PRAKTIKEN

|                         |                                                                                      |                                                                                       |                                                                                       |                                                                                                                                                                                |
|-------------------------|--------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Continuous Delivery     |   |                                                                                       |   | <br>   |
| Iterative Entwicklung   |  |  |  | <br> |
| Komponentenorientierung |  |  |  |                                                                                           |
| Test First              |  |  |  | <br> |

# 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



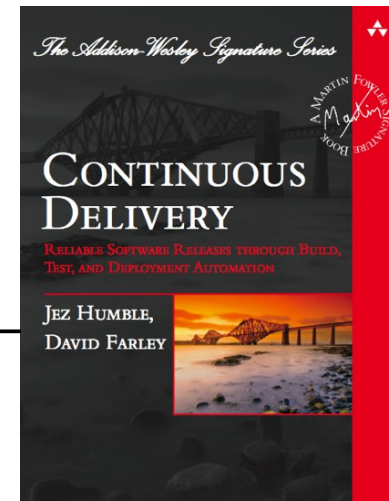
Iterative Entwicklung



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



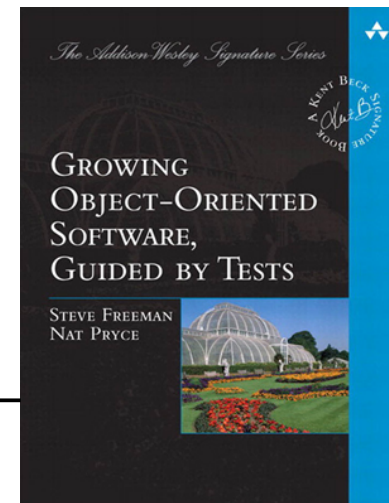
Iterative Entwicklung



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



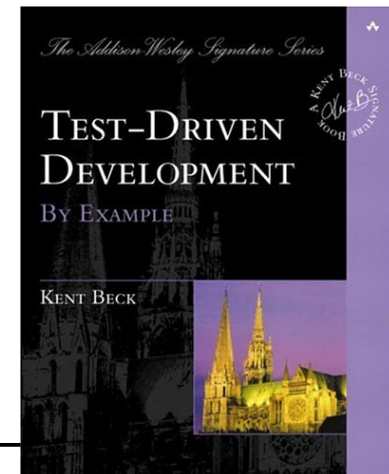
Iterative Entwicklung



Komponentenorientierung



Test First





# Der weiße Grad im Detail

*I'm a Clean Coder!*

## PRINZIPIEN

## PRAKTIKEN

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

# Der weiße Grad im Detail

*I'm a Clean Coder!*

## PRINZIPIEN

## PRAKTIKEN

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



# 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

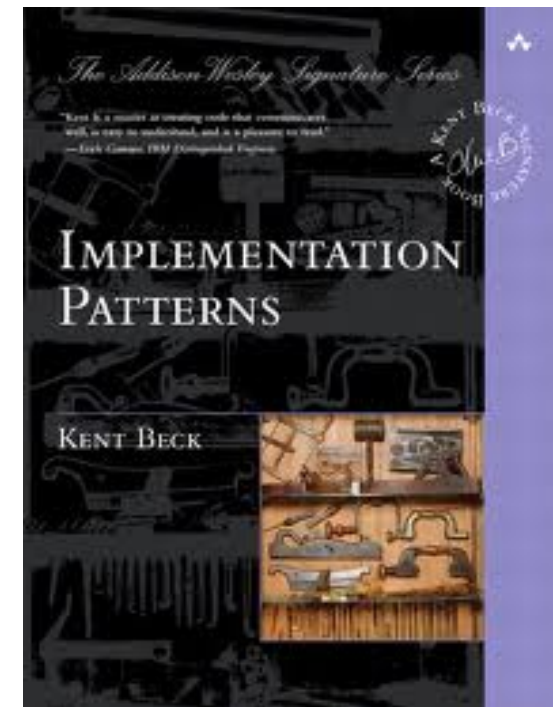
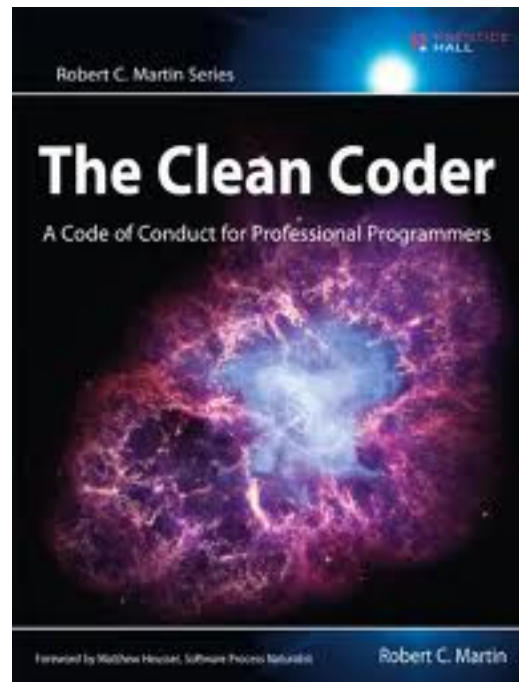
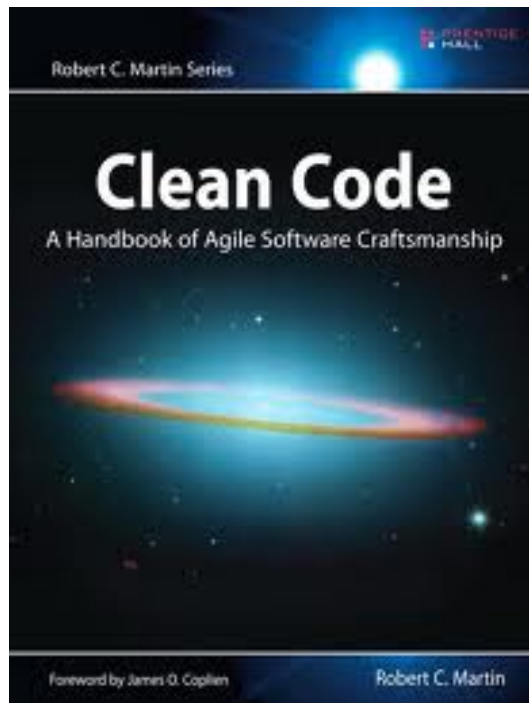
---

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

Community-Seite, Fokus auf .NET

Diskussionsgruppen auf Google und XING

# Literatur direkt zum Thema



# Das Reverse-Prisma

