Individuelle IT-Konzepte
und Softwarelösungen

Gesellschaft für
Informations- und
Kommunikationssysteme mbH

Gesellschaft für
Informations- und
Kommunikationssysteme mbH

Clean Code

Von der Lehre in den Alltag

von **Jörg Vollmer**
und **Reik Oberrath**

27.02.2012

Allgemeines

iks ist IT-Beratungs- und Softwarepartner für Finanzdienstleister, Logistik- und Handelsunternehmen sowie namhafte Unternehmen anderer Branchen. Wir erstellen individuelle IT-Konzepte und Softwarelösungen im Auftrag unserer Kunden.

- **Gründung:** 1989
- **Firmensitz:** Hilden
- **Team:** ca. 90 BeraterInnen
- **Geschäftsführung:** Dipl.-Informatikerin Monika Stoll
- **Prokuristen:** Dipl.-Ing. (FH) Thomas Kondring
  Hartwig Tödter

Gesellschaft für
Informations- und
Kommunikationssysteme mbH

Geschäftsfelder

- **Softwareentwicklung**
  - Java
  - .NET
  - Mobile Applikationen
  - iSeries (RPG / Cobol)
- **IT-Konzepte**
  - Anforderungsanalyse
  - Fachkonzepte
  - Pflichtenhefte
  - DV-Konzepte
- **IT-Beratung**
  - Softwarearchitektur
  - Technologie
  - Methoden
  - Projektmanagement
  - Schulung, Coaching
  - Soziale Applikationen
- **Business Intelligence**
  - MS SQL Server
  - ETL
  - Reporting
  - Analyse

Clean Code

Von der Lehre in den Alltag

Was ist das Ziel der Softwareentwicklung?

Ein Produkt, das
den Auftraggeber
zufrieden stellt!


Und einen zuverlässigen, effektiven Herstellungsprozess!

Das finale Ziel
(hat Selbstzweck)

Das Zwischenziel
(nur Mittel zum Zweck)

Nicht-funktionale Qualitätsmerkmale von Software

korrekt, skalierbar, performant, stabil, effizient, sicher (Security), zuverlässig, gesetzeskonform, gut bedienbar

wirtschaftlich, wartbar, erweiterbar, veränderbar, analysierbar

Der Faktor „Mensch“

Tools & Technologie

Architektur und Implementation

Agenda

- Einleitung
- Ursachen von Dirty-Code
- Die Clean-Code-Developer-Bewegung
- Das Clean-Code-Controlling
- Meinungen zu Clean-Code
- Was ist die Take-Home-Message?

Studie: Das letzte Jahrzehnt

Projekt-Abbrüche von 30% auf 15% gesunken

> 50% aller Projekte überschreiten Zeit und Budget.
Die Wartungskosten sind nach wie vor sehr hoch.

Sind wir 3F: **f**aul, **f**eige und/oder un**f**ähig?

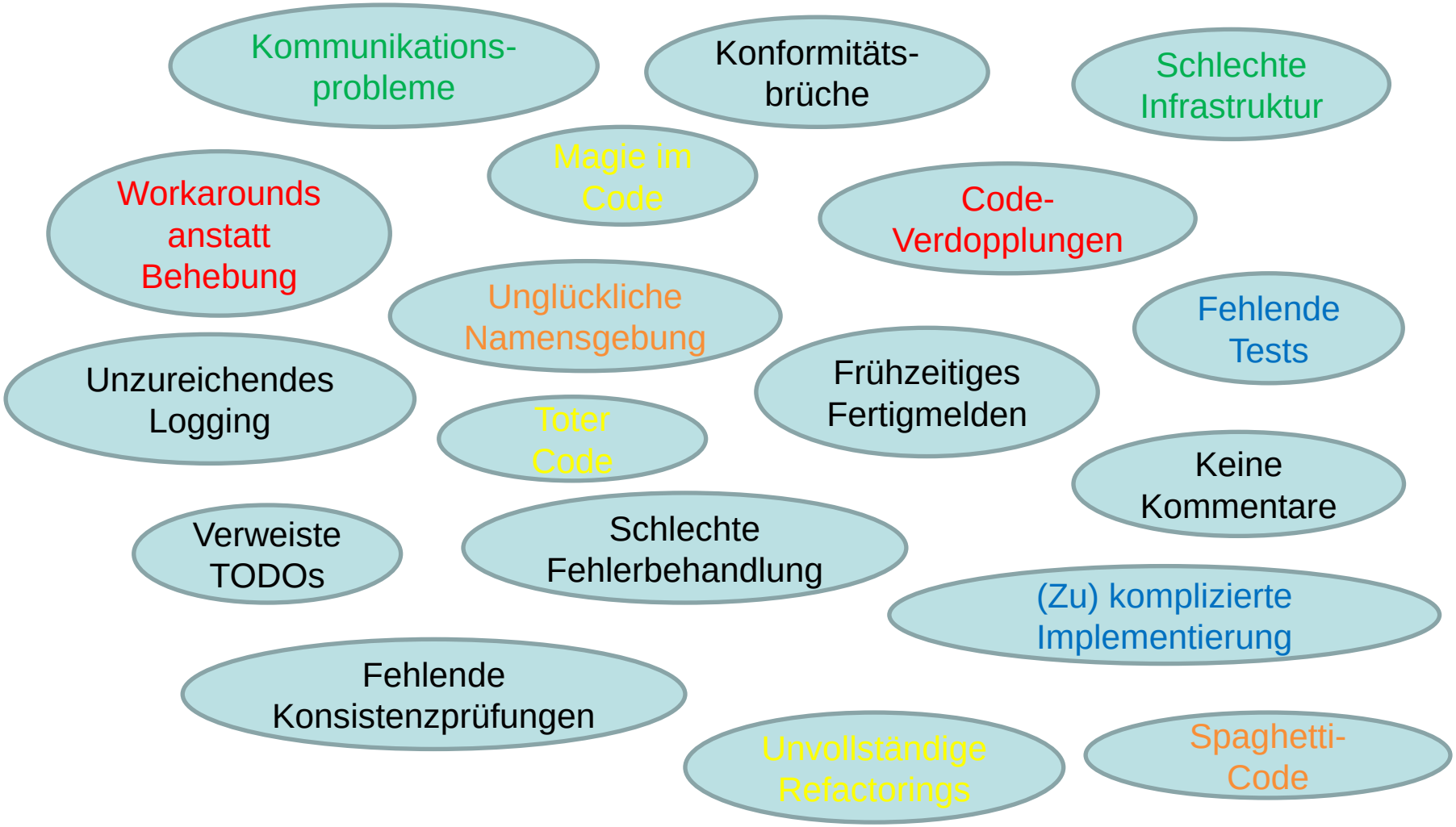
Quellen: Gartner, Roland Berger, Mummert
University Oxford & McKinsey: http://www.cio.de/projektmanagement/machen/2287797/
Ernst & Young: In 48% der Fälle sind technische Gründe die Ursache

Moderne Architekturen


Design Patterns

Der Code

Häufige Unsitten




Läuft

Ein Fall läuft

Die guten Fälle laufen

Fehlerfälle wurden untersucht

„Alle“ Fehlerfälle wurden berücksichtigt

Unit-Tests

Integrationstests

Systemtests

Metriken (statistische Code-Analysen)

Verständlichkeit (menschliche Code-Analyse)

Was bedeutet „läuft“?

DEFINITION OF

1. Erst mal muss es laufen (was, wie, wo)
2. Tipp: DONE bedeutet auch „sauberer Code“

Alleinige „Definition of Done“ reicht nicht, denn wenn der Termin naht…

Agenda

- Ursachen von Dirty-Code
  - Was bedeutet „läuft“?
  - Erst mal muss es laufen
  - Die Gas-Fabrik
  - Die x-te Änderung
  - Broken Windows
  - Konformitätsbrüche
  - Persönliche und organisatorische Defizite
  - Termine

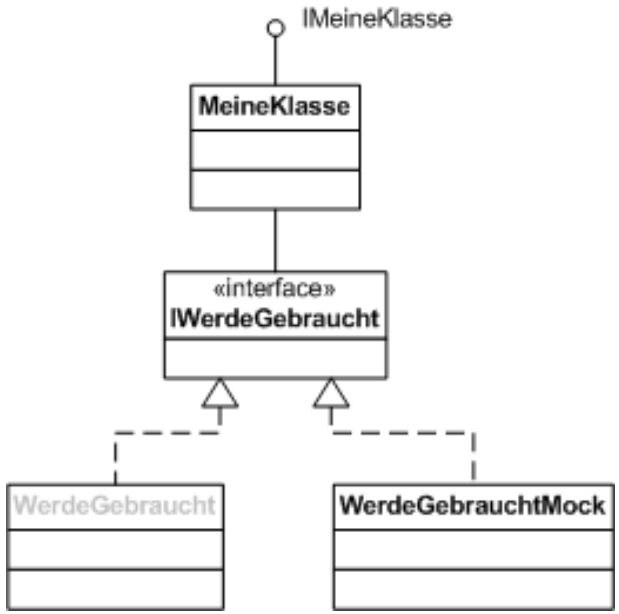
Erst mal muss es laufen …

Warum?

Damit die anderen schon mal damit arbeiten können.

Stimmt nicht (mehr), denn es gibt Mocks.

Interface Segregation Principle (ISP)
Dependency Inversion Principle (DIP)
Inversion of Control Container (ICC)
Test First (TF)



Erst mal muss es laufen…

Warum?

Wir sind agil!

Der Kunde muss die Software möglichst früh kennenlernen.

Projektleiter: „Wie sieht‘s aus? Kann man denn schon etwas zeigen?“

Entwickler: „Wir könnten was vorführen – ist aber erst mal nur ein Schnellschuss, muss auf alle Fälle noch refactored werden…“

Kunde & Projektleiter: „…sieht gut so aus, mehr Funktionen brauchen wir doch gar nicht!“

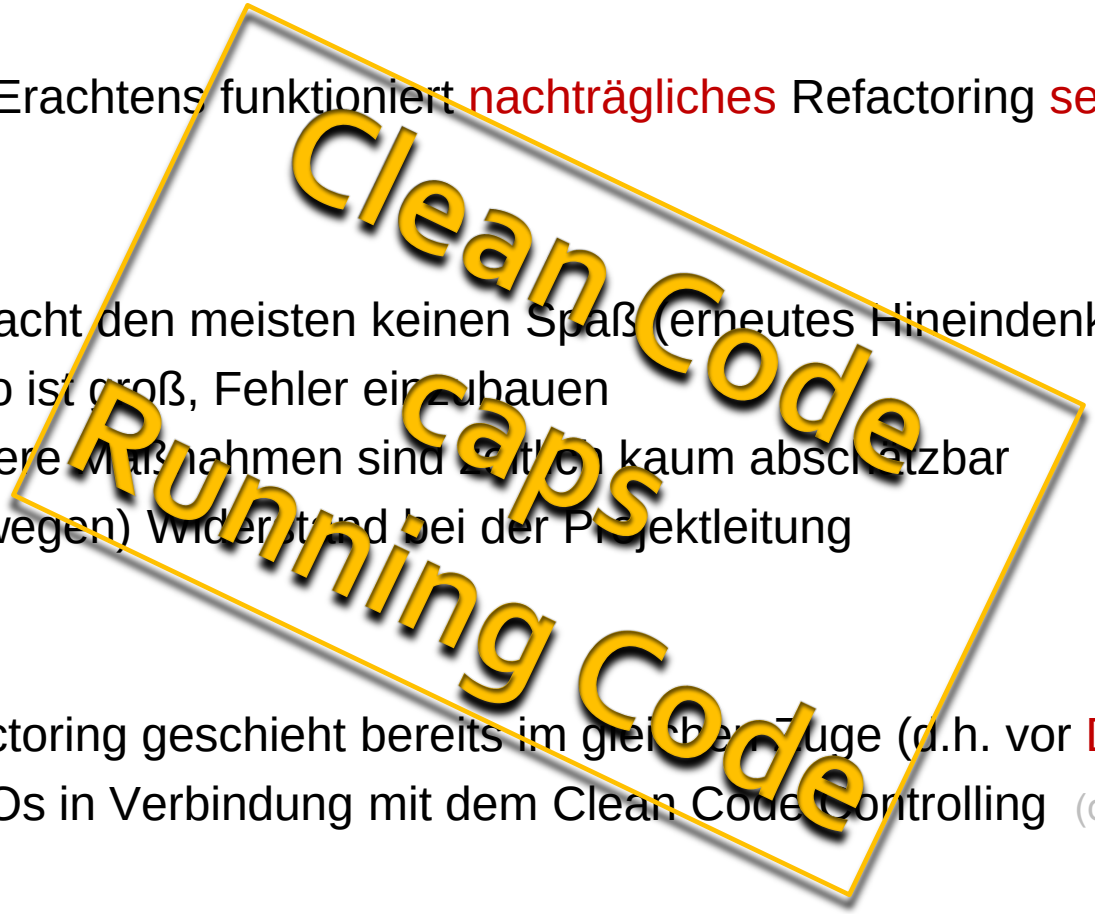
Erst mal muss es laufen….

These:

Unseres Erachtens funktioniert nachträgliches Refactoring seltenst.

Gründe:

- Es macht den meisten keinen Spaß (erneutes Hineindenken)
- Risiko ist groß, Fehler einzubauen
- Größere Maßnahmen sind zeitlich kaum abschätzbar
- (deswegen) Widerstand bei der Projektleitung



Abhilfe:

- Refactoring geschieht bereits im gleichen Zuge (d.h. vor DONE)
- TODOs in Verbindung mit dem Clean Code Controlling (dazu später…)

Agenda

- Ursachen von Dirty-Code
  - Was bedeutet „läuft“?
  - Erst mal muss es laufen
  - Die Gas-Fabrik
  - Die x-te Änderung
  - Broken Windows
  - Konformitätsbrüche
  - Persönliche und organisatorische Defizite
  - Termine

Die Gas-Fabrik

unnötig komplexe Lösung

Ursache: Spieltrieb, unnötige Optimierungen **Abhilfe: KISS, YAGNI**

Die x-te Änderung

Gerade die frühen Software-Module unterliegen dem Desaster der x-ten technischen & fachlichen Änderung.

Abhilfe:

- Erst beginnen, wenn fachlich & technisch alles geklärt ist - realistisch?
- TODOs in Verbindung mit dem Clean Code Controlling (dazu später…)

Broken Windows

Das ist nicht von mir!

Abhilfe: Pfadfinder-Regel

Agenda

- Ursachen von Dirty-Code
  - Was bedeutet „läuft“?
  - Erst mal muss es laufen
  - Die Gas-Fabrik
  - Die x-te Änderung
  - Broken Windows
  - Konformitätsbrüche
  - Persönliche und organisatorische Defizite
  - Termine

Konformitätsbrüche

… kosten in Summe richtig viel Geld!

Ursachen:

- Keine Konventionen oder deren Missachtung
- Unvollständige Refactorings
- Der Reiz der individuellen Lösung
- Scheuklappen
- Einfach keine Lust

Da helfen nur ▼

1. Mentale Einstellung
2. Disziplin
3. Kontrolle

Agenda

- Ursachen von Dirty-Code
  - Was bedeutet „läuft“?
  - Erst mal muss es laufen
  - Die Gas-Fabrik
  - Die x-te Änderung
  - Broken Windows
  - Konformitätsbrüche
  - Persönliche und organisatorische Defizite
  - Termine

Persönliche und organisatorische Defizite

Fortbildungsdefizite

- Unzureichende Weiterbildungsmöglichkeiten
- Mangelndes Interesse

Menschliche Probleme

- Sender-Empfänger-Probleme
- Schlechtes Team-Klima

Organisationsprobleme

- Unklare Rollen, Prozesse, Richtlinien

Motivationsprobleme

- Kein Bewertungssystem für Lob und Tadel

Agenda

- Ursachen von Dirty-Code
  - Was bedeutet „läuft“?
  - Erst mal muss es laufen
  - Die Gas-Fabrik
  - Die x-te Änderung
  - Broken Windows
  - Konformitätsbrüche
  - Persönliche und organisatorische Defizite
  - Termine

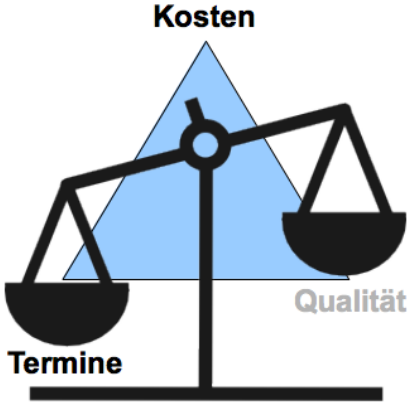
Termine

Projektleiter: „Nur unter Druck entstehen Diamanten.“

Termine haben erfahrungsgemäß negativen Einfluss auf Clean-Code.

Dilemma:

- Termine wird es immer geben,
- Anzahl von Issues/Tickets ist direkt messbar, Qualität hingegen nicht.
- Entwickler sind gehemmt, die Wahrheit zu sagen.



Wie erreichen wir ein Gegengewicht?

- Bedenke stets Clean-Code-Aufwände bei Aufwandsabschätzungen!
- Clean-Code-Development appelliert an die mentale Einstellung.

Wir denken:

Ein *Prozess* muss zusätzlich helfen. → **Clean Code Controlling**

Verhindern von Dirty-Code

- Ursachen von Dirty-Code
  - Was bedeutet „läuft“?
  - Erst mal muss es laufen
  - Die Gas-Fabrik
  - Die x-te Änderung
  - Broken Windows
  - Konformitätsbrüche
  - Persönliche und organisatorische Defizite
  - Termine

→ Prinzipien & Praktiken, Selbstkontrolle & Werte-system

Clean Code Controlling → Teamarbeit, Kommunikation & Temkontrolle

Agenda

- Einleitung
- Ursachen von Dirty-Code
- Die Clean-Code-Developer-Bewegung
- Das Clean Code Controlling
- Meinungen zu Clean-Code
- Was ist die Take-Home-Message?


Mr. Clean

Definition

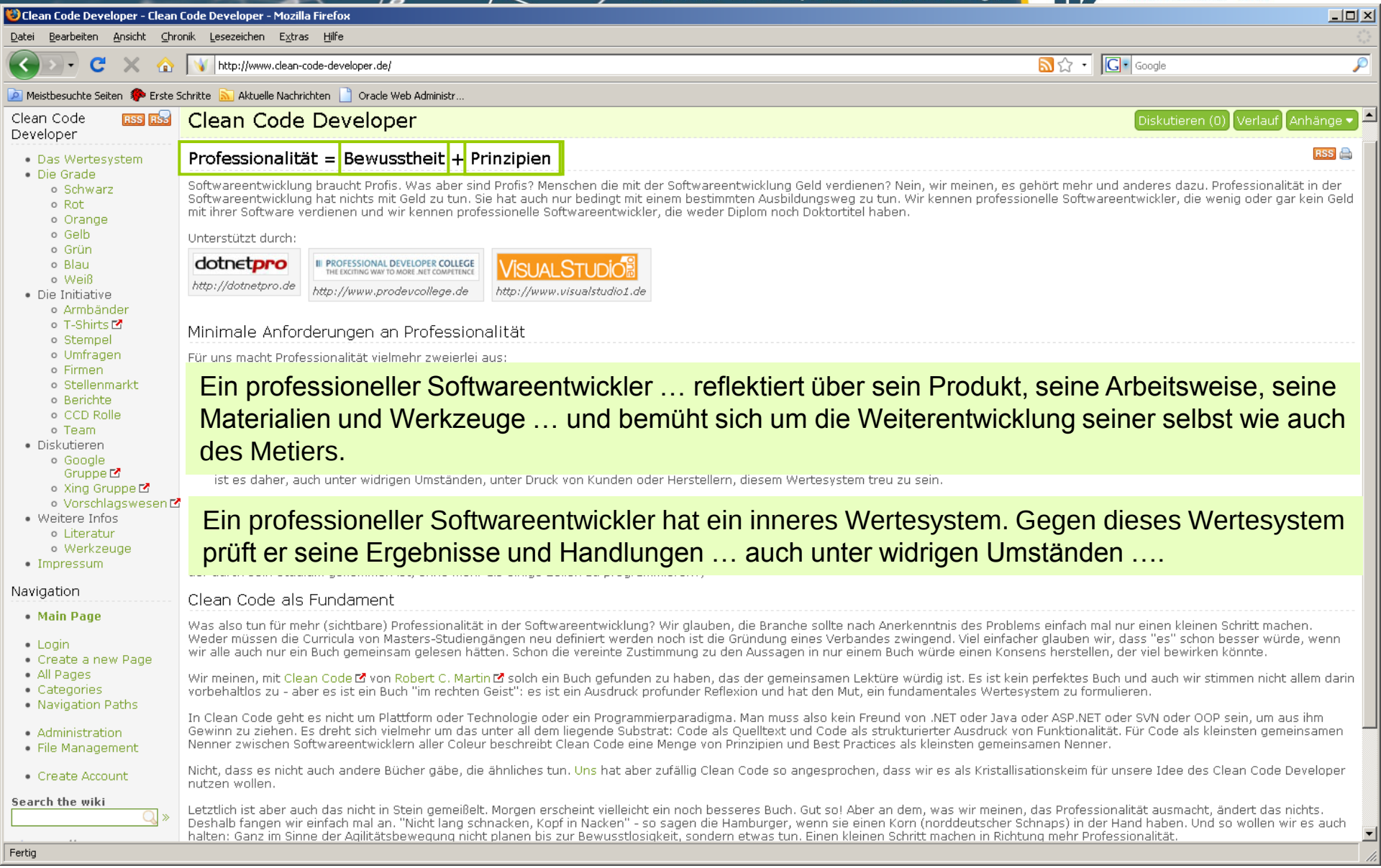
Def. 1: *Clean* ist alles, was intuitiv verständlich ist, also (Quellcode-) Dokumente, Datenstrukturen, Konzepte und Regeln.

Def. 2: *Intuitiv verständlich* ist, was mit wenig Spezialwissen in kurzer Zeit richtig verstanden wird.

Def. 3: Clean ist alles, was effizient ist, also Verfahren, die die Softwareentwicklung beschleunigen.

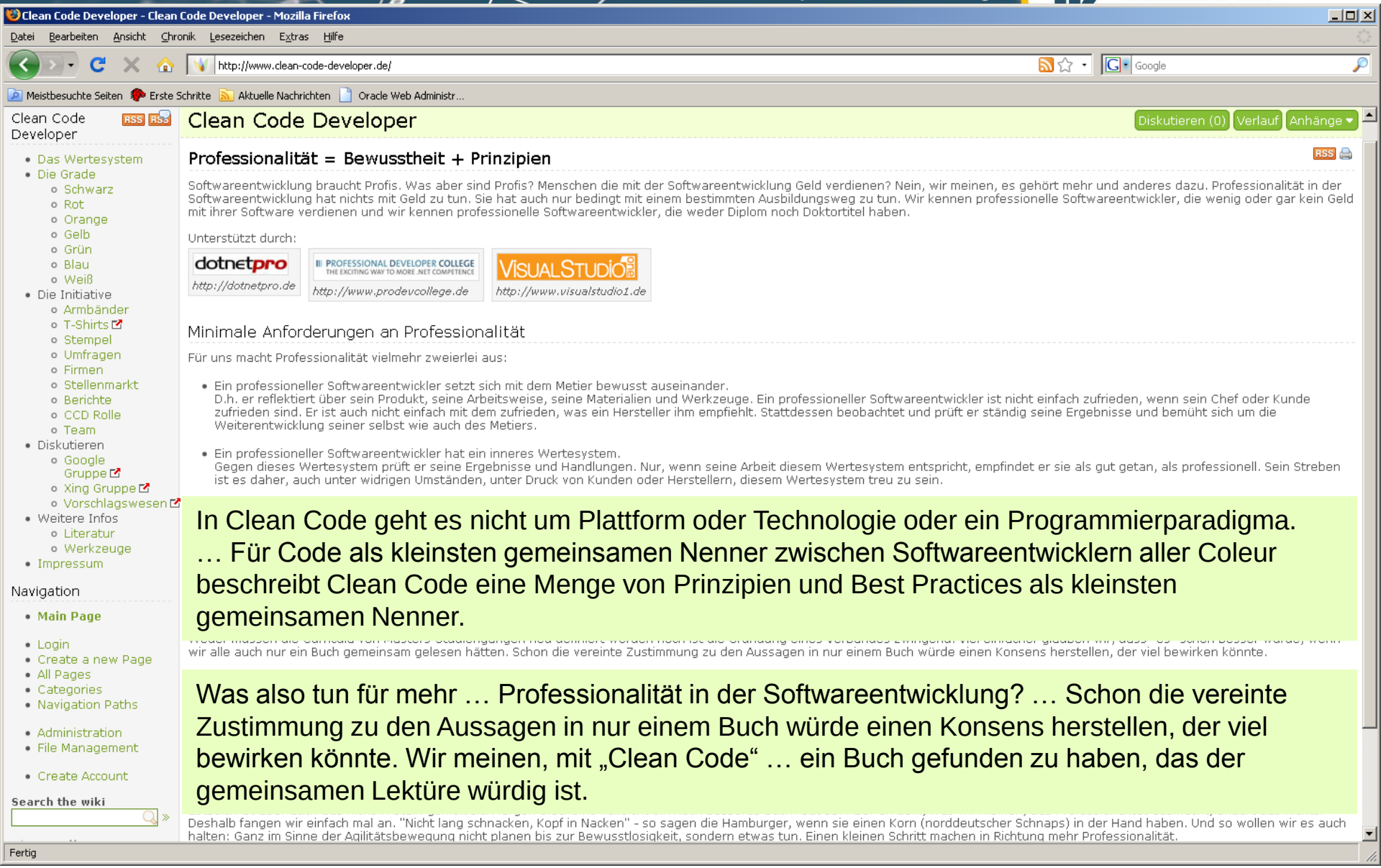
Kernaussagen der Clean-Code-Developer-Bewegung

- Professionalität
- Wertesystem
- Programmieralltag


Clean Code Developer - Clean Code Developer - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://www.clean-code-developer.de/
Google
Meistbesuchte Seiten Erste Schritte Aktuelle Nachrichten Oracle Web Administr...
Clean Code Developer
Das Wertesystem
Die Grade
Schwarz
Rot
Orange
Gelb
Grün
Blau
Weiß
Die Initiative
Armbänder
T-Shirts
Stempel
Umfragen
Firmen
Stellenmarkt
Berichte
CCD Rolle
Team
Diskutieren
Google Gruppe
Xing Gruppe
Vorschlagswesen
Weitere Infos
Literatur
Werkzeuge
Impressum
Navigation
Main Page
Login
Create a new Page
All Pages
Categories
Navigation Paths
Administration
File Management
Create Account
Search the wiki
Clean Code Developer
Diskutieren (0) Verlauf Anhänge
Professionalität = Bewusstheit + Prinzipien
Softwareentwicklung braucht Profis. Was aber sind Profis? Menschen die mit der Softwareentwicklung Geld verdienen? Nein, wir meinen, es gehört mehr und anderes dazu. Professionalität in der Softwareentwicklung hat nichts mit Geld zu tun. Sie hat auch nur bedingt mit einem bestimmten Ausbildungsweg zu tun. Wir kennen professionelle Softwareentwickler, die wenig oder gar kein Geld mit ihrer Software verdienen und wir kennen professionelle Softwareentwickler, die weder Diplom noch Doktortitel haben.
Unterstützt durch:
dotnetpro
http://dotnetpro.de
PROFESSIONAL DEVELOPER COLLEGE
THE EXCITING WAY TO MORE .NET COMPETENCE
http://www.prodevcollege.de
VISUAL STUDIO
http://www.visualstudio1.de
Minimale Anforderungen an Professionalität
Für uns macht Professionalität vielmehr zweierlei aus:
Ein professioneller Softwareentwickler … reflektiert über sein Produkt, seine Arbeitsweise, seine Materialien und Werkzeuge … und bemüht sich um die Weiterentwicklung seiner selbst wie auch des Metiers.
ist es daher, auch unter widrigen Umständen, unter Druck von Kunden oder Herstellern, diesem Wertesystem treu zu sein.
Ein professioneller Softwareentwickler hat ein inneres Wertesystem. Gegen dieses Wertesystem prüft er seine Ergebnisse und Handlungen … auch unter widrigen Umständen ….
Clean Code als Fundament
Was also tun für mehr (sichtbare) Professionalität in der Softwareentwicklung? Wir glauben, die Branche sollte nach Anerkenntnis des Problems einfach mal nur einen kleinen Schritt machen. Weder müssen die Curricula von Masters-Studiengängen neu definiert werden noch ist die Gründung eines Verbandes zwingend. Viel einfacher glauben wir, dass "es" schon besser würde, wenn wir alle auch nur ein Buch gemeinsam gelesen hätten. Schon die vereinte Zustimmung zu den Aussagen in nur einem Buch würde einen Konsens herstellen, der viel bewirken könnte.
Wir meinen, mit Clean Code von Robert C. Martin solch ein Buch gefunden zu haben, das der gemeinsamen Lektüre würdig ist. Es ist kein perfektes Buch und auch wir stimmen nicht allem darin vorbehaltlos zu - aber es ist ein Buch "im rechten Geist": es ist ein Ausdruck profunder Reflexion und hat den Mut, ein fundamentales Wertesystem zu formulieren.
In Clean Code geht es nicht um Plattform oder Technologie oder ein Programmierparadigma. Man muss also kein Freund von .NET oder Java oder ASP.NET oder SVN oder OOP sein, um aus ihm Gewinn zu ziehen. Es dreht sich vielmehr um das unter all dem liegende Substrat: Code als Quelltext und Code als strukturierter Ausdruck von Funktionalität. Für Code als kleinsten gemeinsamen Nenner zwischen Softwareentwicklern aller Coleur beschreibt Clean Code eine Menge von Prinzipien und Best Practices als kleinsten gemeinsamen Nenner.
Nicht, dass es nicht auch andere Bücher gäbe, die ähnliches tun. Uns hat aber zufällig Clean Code so angesprochen, dass wir es als Kristallisationskeim für unsere Idee des Clean Code Developer nutzen wollen.
Letztlich ist aber auch das nicht in Stein gemeißelt. Morgen erscheint vielleicht ein noch besseres Buch. Gut so! Aber an dem, was wir meinen, das Professionalität ausmacht, ändert das nichts. Deshalb fangen wir einfach mal an. "Nicht lang schnacken, Kopf in Nacken" - so sagen die Hamburger, wenn sie einen Korn (norddeutscher Schnaps) in der Hand haben. Und so wollen wir es auch halten: Ganz im Sinne der Agilitätsbewegung nicht planen bis zur Bewusstlosigkeit, sondern etwas tun. Einen kleinen Schritt machen in Richtung mehr Professionalität.
Fertig

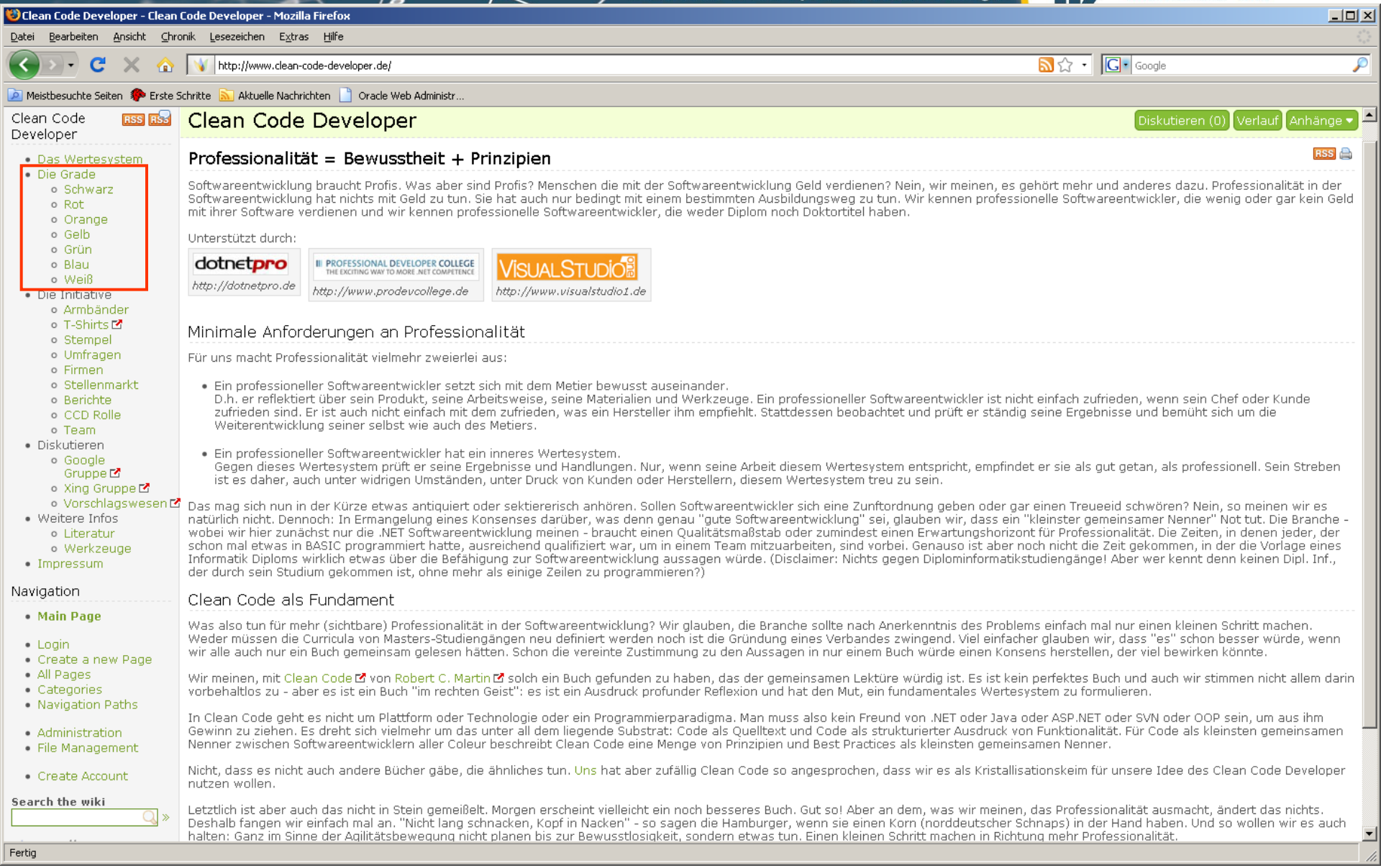
Kernaussagen der Clean-Code-Developer-Bewegung

- Professionalität — Ständige Selbstkontrolle durch konsequente Anwendung des inneren Wertesystems
- Wertesystem
- Programmieralltag


Clean Code Developer - Clean Code Developer - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://www.clean-code-developer.de/
Google
Meistbesuchte Seiten Erste Schritte Aktuelle Nachrichten Oracle Web Administr...
Clean Code Developer
Das Wertesystem
Die Grade
Schwarz
Rot
Orange
Gelb
Grün
Blau
Weiß
Die Initiative
Armbänder
T-Shirts
Stempel
Umfragen
Firmen
Stellenmarkt
Berichte
CCD Rolle
Team
Diskutieren
Google Gruppe
Xing Gruppe
Vorschlagswesen
Weitere Infos
Literatur
Werkzeuge
Impressum
Navigation
Main Page
Login
Create a new Page
All Pages
Categories
Navigation Paths
Administration
File Management
Create Account
Search the wiki
Clean Code Developer
Diskutieren (0)
Verlauf
Anhänge
Professionalität = Bewusstheit + Prinzipien
Softwareentwicklung braucht Profis. Was aber sind Profis? Menschen die mit der Softwareentwicklung Geld verdienen? Nein, wir meinen, es gehört mehr und anderes dazu. Professionalität in der Softwareentwicklung hat nichts mit Geld zu tun. Sie hat auch nur bedingt mit einem bestimmten Ausbildungsweg zu tun. Wir kennen professionelle Softwareentwickler, die wenig oder gar kein Geld mit ihrer Software verdienen und wir kennen professionelle Softwareentwickler, die weder Diplom noch Doktortitel haben.
Unterstützt durch:
dotnetpro
http://dotnetpro.de
PROFESSIONAL DEVELOPER COLLEGE
THE EXCITING WAY TO MORE .NET COMPETENCE
http://www.prodevcollege.de
VISUAL STUDIO
http://www.visualstudio1.de
Minimale Anforderungen an Professionalität
Für uns macht Professionalität vielmehr zweierlei aus:
Ein professioneller Softwareentwickler setzt sich mit dem Metier bewusst auseinander.
D.h. er reflektiert über sein Produkt, seine Arbeitsweise, seine Materialien und Werkzeuge. Ein professioneller Softwareentwickler ist nicht einfach zufrieden, wenn sein Chef oder Kunde zufrieden sind. Er ist auch nicht einfach mit dem zufrieden, was ein Hersteller ihm empfiehlt. Stattdessen beobachtet und prüft er ständig seine Ergebnisse und bemüht sich um die Weiterentwicklung seiner selbst wie auch des Metiers.
Ein professioneller Softwareentwickler hat ein inneres Wertesystem.
Gegen dieses Wertesystem prüft er seine Ergebnisse und Handlungen. Nur, wenn seine Arbeit diesem Wertesystem entspricht, empfindet er sie als gut getan, als professionell. Sein Streben ist es daher, auch unter widrigen Umständen, unter Druck von Kunden oder Herstellern, diesem Wertesystem treu zu sein.
In Clean Code geht es nicht um Plattform oder Technologie oder ein Programmierparadigma. … Für Code als kleinsten gemeinsamen Nenner zwischen Softwareentwicklern aller Coleur beschreibt Clean Code eine Menge von Prinzipien und Best Practices als kleinsten gemeinsamen Nenner.
wir alle auch nur ein Buch gemeinsam gelesen hätten. Schon die vereinte Zustimmung zu den Aussagen in nur einem Buch würde einen Konsens herstellen, der viel bewirken könnte.
Was also tun für mehr … Professionalität in der Softwareentwicklung? … Schon die vereinte Zustimmung zu den Aussagen in nur einem Buch würde einen Konsens herstellen, der viel bewirken könnte. Wir meinen, mit „Clean Code“ … ein Buch gefunden zu haben, das der gemeinsamen Lektüre würdig ist.
Deshalb fangen wir einfach mal an. "Nicht lang schnacken, Kopf in Nacken" - so sagen die Hamburger, wenn sie einen Korn (norddeutscher Schnaps) in der Hand haben. Und so wollen wir es auch halten: Ganz im Sinne der Agilitätsbewegung nicht planen bis zur Bewusstlosigkeit, sondern etwas tun. Einen kleinen Schritt machen in Richtung mehr Professionalität.
Fertig

Kernaussagen der Clean-Code-Developer-Bewegung

- Professionalität — Ständige Selbstkontrolle durch konsequente Anwendung des inneren Wertesystems
- Wertesystem — Das Buch „Clean Code“ ist wert, als allgemeingültiges Wertesystem anerkannt zu werden.
- Programmieralltag


Clean Code Developer - Clean Code Developer - Mozilla Firefox
Datei Bearbeiten Ansicht Chronik Lesezeichen Extras Hilfe
http://www.clean-code-developer.de/
Google
Meistbesuchte Seiten
Erste Schritte
Aktuelle Nachrichten
Oracle Web Administr...
Clean Code Developer
Clean Code Developer
Diskutieren (0)
Verlauf
Anhänge
Professionalität = Bewusstheit + Prinzipien
Softwareentwicklung braucht Profis. Was aber sind Profis? Menschen die mit der Softwareentwicklung Geld verdienen? Nein, wir meinen, es gehört mehr und anderes dazu. Professionalität in der Softwareentwicklung hat nichts mit Geld zu tun. Sie hat auch nur bedingt mit einem bestimmten Ausbildungsweg zu tun. Wir kennen professionelle Softwareentwickler, die wenig oder gar kein Geld mit ihrer Software verdienen und wir kennen professionelle Softwareentwickler, die weder Diplom noch Doktortitel haben.
Unterstützt durch:
dotnetpro
http://dotnetpro.de
PROFESSIONAL DEVELOPER COLLEGE
THE EXCITING WAY TO MORE .NET COMPETENCE
http://www.prodevcollege.de
VISUAL STUDIO
http://www.visualstudio1.de
Minimale Anforderungen an Professionalität
Für uns macht Professionalität vielmehr zweierlei aus:
Ein professioneller Softwareentwickler setzt sich mit dem Metier bewusst auseinander.
D.h. er reflektiert über sein Produkt, seine Arbeitsweise, seine Materialien und Werkzeuge. Ein professioneller Softwareentwickler ist nicht einfach zufrieden, wenn sein Chef oder Kunde zufrieden sind. Er ist auch nicht einfach mit dem zufrieden, was ein Hersteller ihm empfiehlt. Stattdessen beobachtet und prüft er ständig seine Ergebnisse und bemüht sich um die Weiterentwicklung seiner selbst wie auch des Metiers.
Ein professioneller Softwareentwickler hat ein inneres Wertesystem.
Gegen dieses Wertesystem prüft er seine Ergebnisse und Handlungen. Nur, wenn seine Arbeit diesem Wertesystem entspricht, empfindet er sie als gut getan, als professionell. Sein Streben ist es daher, auch unter widrigen Umständen, unter Druck von Kunden oder Herstellern, diesem Wertesystem treu zu sein.
Das mag sich nun in der Kürze etwas antiquiert oder sektiererisch anhören. Sollen Softwareentwickler sich eine Zunftordnung geben oder gar einen Treueeid schwören? Nein, so meinen wir es natürlich nicht. Dennoch: In Ermangelung eines Konsenses darüber, was denn genau "gute Softwareentwicklung" sei, glauben wir, dass ein "kleinster gemeinsamer Nenner" Not tut. Die Branche - wobei wir hier zunächst nur die .NET Softwareentwicklung meinen - braucht einen Qualitätsmaßstab oder zumindest einen Erwartungshorizont für Professionalität. Die Zeiten, in denen jeder, der schon mal etwas in BASIC programmiert hatte, ausreichend qualifiziert war, um in einem Team mitzuarbeiten, sind vorbei. Genauso ist aber noch nicht die Zeit gekommen, in der die Vorlage eines Informatik Diploms wirklich etwas über die Befähigung zur Softwareentwicklung aussagen würde. (Disclaimer: Nichts gegen Diplominformatikstudiengänge! Aber wer kennt denn keinen Dipl. Inf., der durch sein Studium gekommen ist, ohne mehr als einige Zeilen zu programmieren?)
Clean Code als Fundament
Was also tun für mehr (sichtbare) Professionalität in der Softwareentwicklung? Wir glauben, die Branche sollte nach Anerkenntnis des Problems einfach mal nur einen kleinen Schritt machen. Weder müssen die Curricula von Masters-Studiengängen neu definiert werden noch ist die Gründung eines Verbandes zwingend. Viel einfacher glauben wir, dass "es" schon besser würde, wenn wir alle auch nur ein Buch gemeinsam gelesen hätten. Schon die vereinte Zustimmung zu den Aussagen in nur einem Buch würde einen Konsens herstellen, der viel bewirken könnte.
Wir meinen, mit Clean Code von Robert C. Martin solch ein Buch gefunden zu haben, das der gemeinsamen Lektüre würdig ist. Es ist kein perfektes Buch und auch wir stimmen nicht allem darin vorbehaltlos zu - aber es ist ein Buch "im rechten Geist": es ist ein Ausdruck profunder Reflexion und hat den Mut, ein fundamentales Wertesystem zu formulieren.
In Clean Code geht es nicht um Plattform oder Technologie oder ein Programmierparadigma. Man muss also kein Freund von .NET oder Java oder ASP.NET oder SVN oder OOP sein, um aus ihm Gewinn zu ziehen. Es dreht sich vielmehr um das unter all dem liegende Substrat: Code als Quelltext und Code als strukturierter Ausdruck von Funktionalität. Für Code als kleinsten gemeinsamen Nenner zwischen Softwareentwicklern aller Coleur beschreibt Clean Code eine Menge von Prinzipien und Best Practices als kleinsten gemeinsamen Nenner.
Nicht, dass es nicht auch andere Bücher gäbe, die ähnliches tun. Uns hat aber zufällig Clean Code so angesprochen, dass wir es als Kristallisationskeim für unsere Idee des Clean Code Developer nutzen wollen.
Letztlich ist aber auch das nicht in Stein gemeißelt. Morgen erscheint vielleicht ein noch besseres Buch. Gut so! Aber an dem, was wir meinen, das Professionalität ausmacht, ändert das nichts. Deshalb fangen wir einfach mal an. "Nicht lang schnacken, Kopf in Nacken" - so sagen die Hamburger, wenn sie einen Korn (norddeutscher Schnaps) in der Hand haben. Und so wollen wir es auch halten: Ganz im Sinne der Agilitätsbewegung nicht planen bis zur Bewusstlosigkeit, sondern etwas tun. Einen kleinen Schritt machen in Richtung mehr Professionalität.
Das Wertesystem
Die Grade
Schwarz
Rot
Orange
Gelb
Grün
Blau
Weiß
Die Initiative
Armbänder
T-Shirts
Stempel
Umfragen
Firmen
Stellenmarkt
Berichte
CCD Rolle
Team
Diskutieren
Google Gruppe
Xing Gruppe
Vorschlagswesen
Weitere Infos
Literatur
Werkzeuge
Impressum
Navigation
Main Page
Login
Create a new Page
All Pages
Categories
Navigation Paths
Administration
File Management
Create Account
Search the wiki
Fertig

Die Clean-Code-Grade:

	Schwarz	Rot	Orange	Gelb	Grün	Blau	Weiss
Prinzipien		- DRY …	- SLA …	- DIP …	- Open Closed Principle …	- YAGNI …	
Praktiken		- Pfadfinderregel …	- Issue Tracking …	- Automatisierte Unit Tests …	- Continuous Integration …	- Continuous Delivery …	

Prinzipien	- Don´t Repeat Yourself (DRY) - Keep it simple, stupid (KISS) - Vorsicht vor Optimierungen! (VvO) - Favour Composition over Inheritance (FCoI)
Praktiken	- Die Pfadfinderregel beachten - Root Cause Analysis - Ein Versionskontrollsystem einsetzen - Einfache Refaktorisierungsmuster anwenden - Täglich reflektieren

Prinzipien	- Single Level of Abstraction (SLA) - Single Responsibility Principle (SRP) - Separation of Concerns (SoC) - Source Code Conventions (SCC)
Praktiken	- Issue Tracking - Automatisierte Integrationstests - Lesen, Lesen, Lesen - Reviews

Prinzipien	- Interface Segregation Principle (ISP) - Dependency Inversion Principle (DIP) - Liskov Substitution Principle (LSP) - Principle of Least Astonishment (PLA) - Information Hiding Principle (IHP)
Praktiken	- Automatisierte Unit Tests - Mockups (Testattrappen) - Code Coverage Analyse - Teilnahme an Fachveranstaltungen - Komplexe Refaktorisierungen

Prinzipien	- Open Closed Principle (OCP) - Tell, don´t ask (Tda) - Law of Demeter (LoD)
Praktiken	- Continuous Integration - Statische Codeanalyse (Metriken) - Inversion of Control Container - Erfahrung weitergeben - Messen von Fehlern

Prinzipien					- Implementation spiegelt Entwurf (IsE) - You Ain´t Gonna Need It (YAGNI) - Entwurf und Implementation überlappen nicht (EIün)	
Praktiken					- Continuous Delivery (CD) - Iterative Entwicklung - Komponentenorientierung - Test first (TF)	

Kernaussagen der Clean-Code-Developer-Bewegung

- Professionalität — Ständige Selbstkontrolle durch konsequente Anwendung des inneren Wertesystems
- Wertesystem — Das Buch „Clean Code“ ist wert, als allgemeingültiges Wertesystem anerkannt zu werden.
- Programmieralltag — Das CCD-Grade-System hilft, das innere Wertesystem aktiv einzusetzen und die mentale CCD-Einstellung zu verinnerlichen.

Definition

Def. 4: Clean Code ist ein Buch mit einem umfassenden Regelwerk, das als gemeinsames Wertesystem (Best Practices) gilt.

Def. 5: Clean Code ist die mentale Einstellung, das gemeinsame Wertesystem im Programmieralltag bewusst anzuwenden (der „Geist“ der Clean-Code-Developer-Bewegung).

Agenda

- Einleitung
- Ursachen von Dirty-Code
- Die Clean-Code-Developer-Bewegung
- Das Clean Code Controlling
- Meinungen zu Clean-Code
- Was ist die Take-Home-Message?

Verhindern von Dirty-Code

- Ursachen von Dirty-Code
  - Was bedeutet „läuft“?
  - Erst mal muss es laufen
  - Die Gas-Fabrik
  - Die x-te Änderung
  - Broken Windows
  - Konformitätsbrüche
  - Persönliche und organisatorische Defizite
  - Termine

→ Prinzipien & Praktiken, Selbstkontrolle & Werte-system

Clean Code Controlling

→ Teamarbeit, Kommunikation & Teamkontrolle

Das Clean Code Controlling

Warum?

- Gute Vorsätze schwinden schnell → Verrottung
- Unterschiedliche Meinungen → Wildwuchs
- Mittel gegen die „Durchsetzungsstarken“ (die Stillen sind oft besser)
- Motivation durch Anerkennung, Kontrolle auf Augenhöhe

Das Clean Code Controlling

Basis-Konsens

§1 *ALLE Teammitglieder (inkl. Projektleitung) verpflichten sich, Prozesse anzuerkennen, die für ALLE gleichermaßen bindend sind.*

§2 *Diese Prozesse können bei Bedarf jederzeit angepasst werden.*

§3 *Als erstes muss ein effizientes Verfahren zur Entscheidungsfindung etabliert werden.*

Das Clean Code Controlling (Legislative)

Methoden zur Entscheidungsfindung

- Vorgabe des Projektleiters
- Mehrheitsbeschluss
- Minimierung des durchschn. Widerstands
  - Vetoabfrage
  - Konsensrunde
  - Thumb-Voting

„Konsens bedeutet die Übereinstimmung von Menschen hinsichtlich einer Thematik ohne verdeckten oder offenen Widerspruch.“ - Wikipedia

Das Clean Code Controlling (Legislative)

- Was bedeutet für uns DONE?
- Wie gehen wir mit unfertigem Code um?

  Code Tagging:

  FIXMEs, TODOs und REFACTORE_MEs:

  ```
  FIXME jvo von rob 20.02.2011: Fall kunde == null fehlt
  ```

  DONE ⇨ FIXMEs raus, TODOs in den Issue-Tracker
- Wie ist unser Umgang mit „fremdem Eigentum“?
- Was ist für uns clean?

Das Clean Code Controlling

Wege zur Einigung im Team

- Daily Standups (Scrum-ähnlich)
- Regelmäßige Team-Reviews
- Retrospektiven
- Coding-Notes kommunizieren (z.B. E-Mail an Team-Verteiler)
- Pair-Programming von Erfahrenen und Unerfahrenen
- 4  -Reviews

Das Clean Code Controlling

4  -Review

Am Ende der Entwicklung eines Software-Teils sucht der Autor nach einem Teammitglied, das als *Reviewer* dient. Dieser prüft:

- Funktionelle Korrektheit
- Ausreichende Testabdeckung
- FIXME- / TODO-Einträge
- Einhaltung der im Team abgestimmten Clean-Code-Maßnahmen

Erst nach dem OK des Reviewers gilt der Teil als **DONE!**

Das Clean Code Controlling (Judikative)

Die 1€-Regel

Ein Euro wird eingezogen bei Vergehen gegen zuvor abgestimmte Regeln.

Beispiele:

- Build brechen und nach Hause gehen,
- Tests brechen und sich nicht um deren Behebung bemühen,
- Backend-Änderungen ohne Client-Anpassung,
- Domain-Änderungen ohne DB-Skript-Anpassung,
- etc.

Was hilft wann?

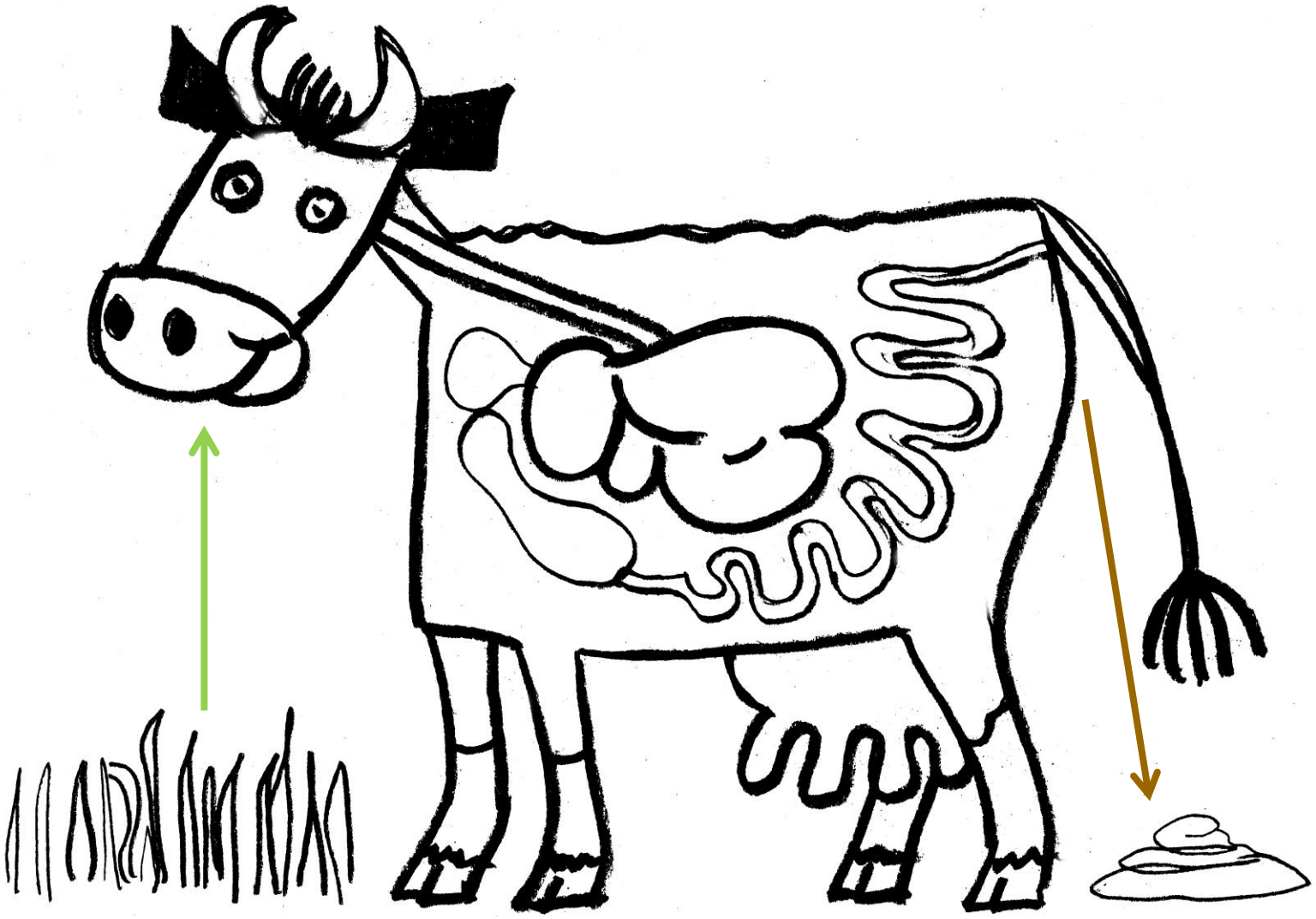
Dirty-Code-Ursachen	Clean Code Dev.	Clean Code Controlling
Unklare DOD		DOD definieren
Erst mal laufen	Test first	DOD anwenden
Gas Factory	KISS, YAGNI, VvO	Team-Reviews, 4iReviews
X-te Änderung		Code Tagging
Broken Windows	Pfadfinderregel	Code Tagging
Konformitätsbrüche	Reviews, ER, SCC	4iReviews
Fortbildung	3L, TAF	Team-Reviews
Kommunikation	Erfahrung weitergeben	Wege zur Einigung im Team
Organisation		Rollen & Prozesse definieren
Motivation	Bewusstsein	Anerkennung, 1€-Regel
Termine	Mentale Einstellung	DOD

Agenda

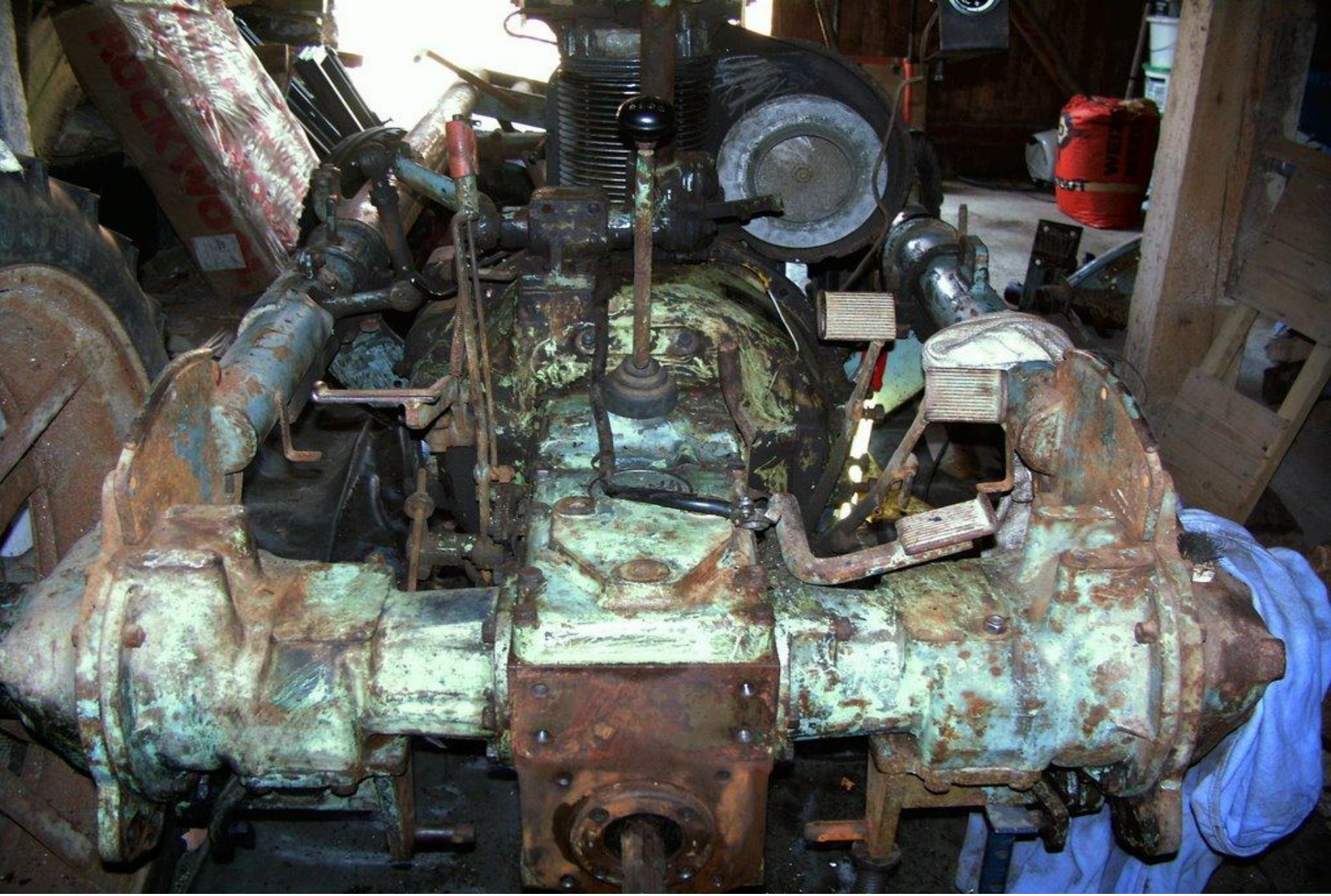
- Einleitung
- Ursachen von Dirty-Code
- Die Clean-Code-Developer-Bewegung
- Das Clean Code Controlling
- Meinungen zu Clean-Code
- Was ist die Take-Home-Message?

Clean-Code ist gut, aber praxisfern – eher etwas für Ästheten!

Clean-Code ist gut, man kann es aber leicht übertreiben!



Ein Organismus verträgt nur ein begrenztes Maß an Schadstoffen.



Clean-Code ist was für Spießer – wichtig ist nur, dass es läuft!

Sauberer Code - Hohe Testabdeckung

Produktionscode

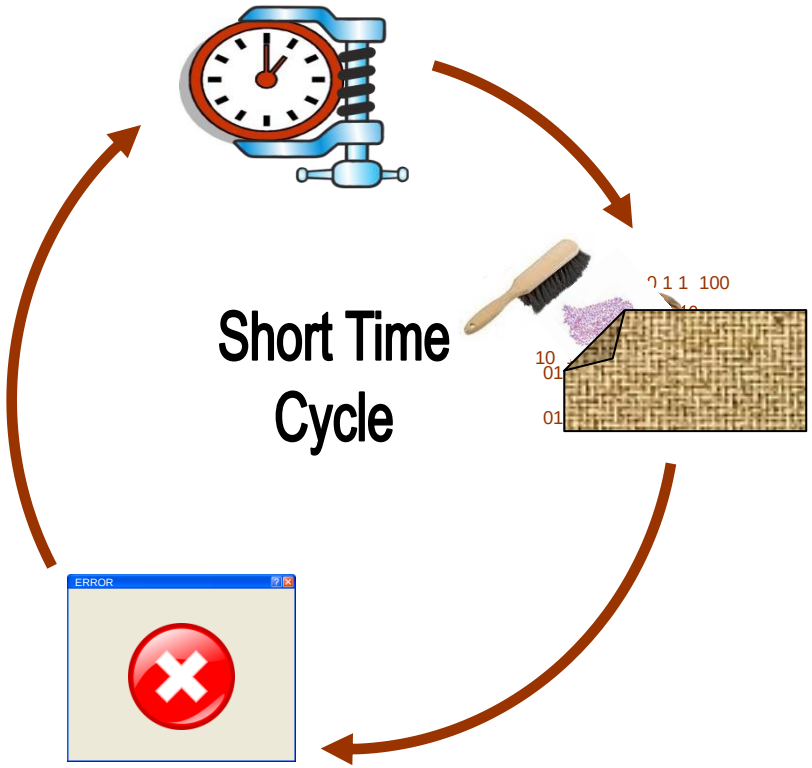
Testabdeckung

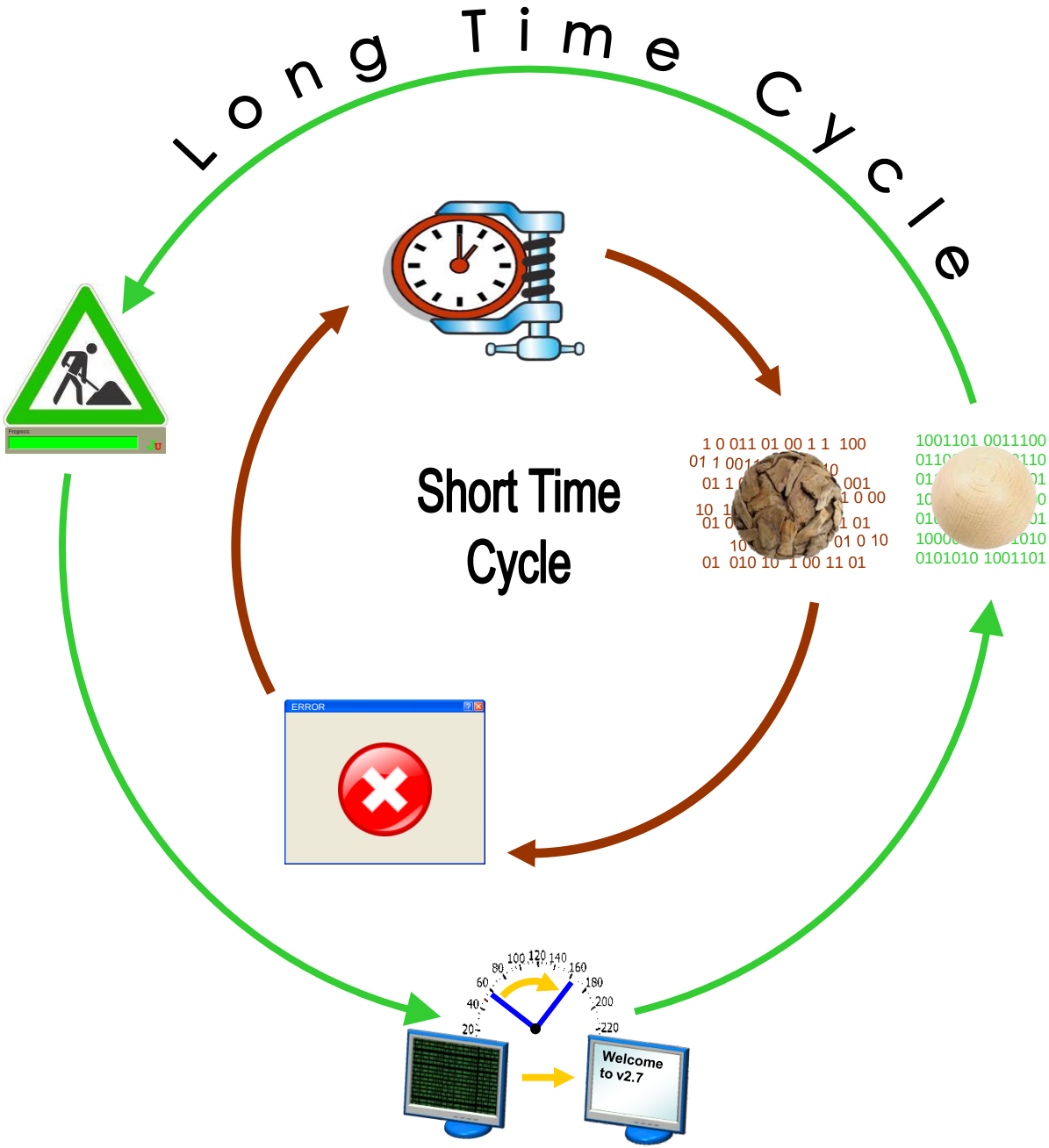
Testcode

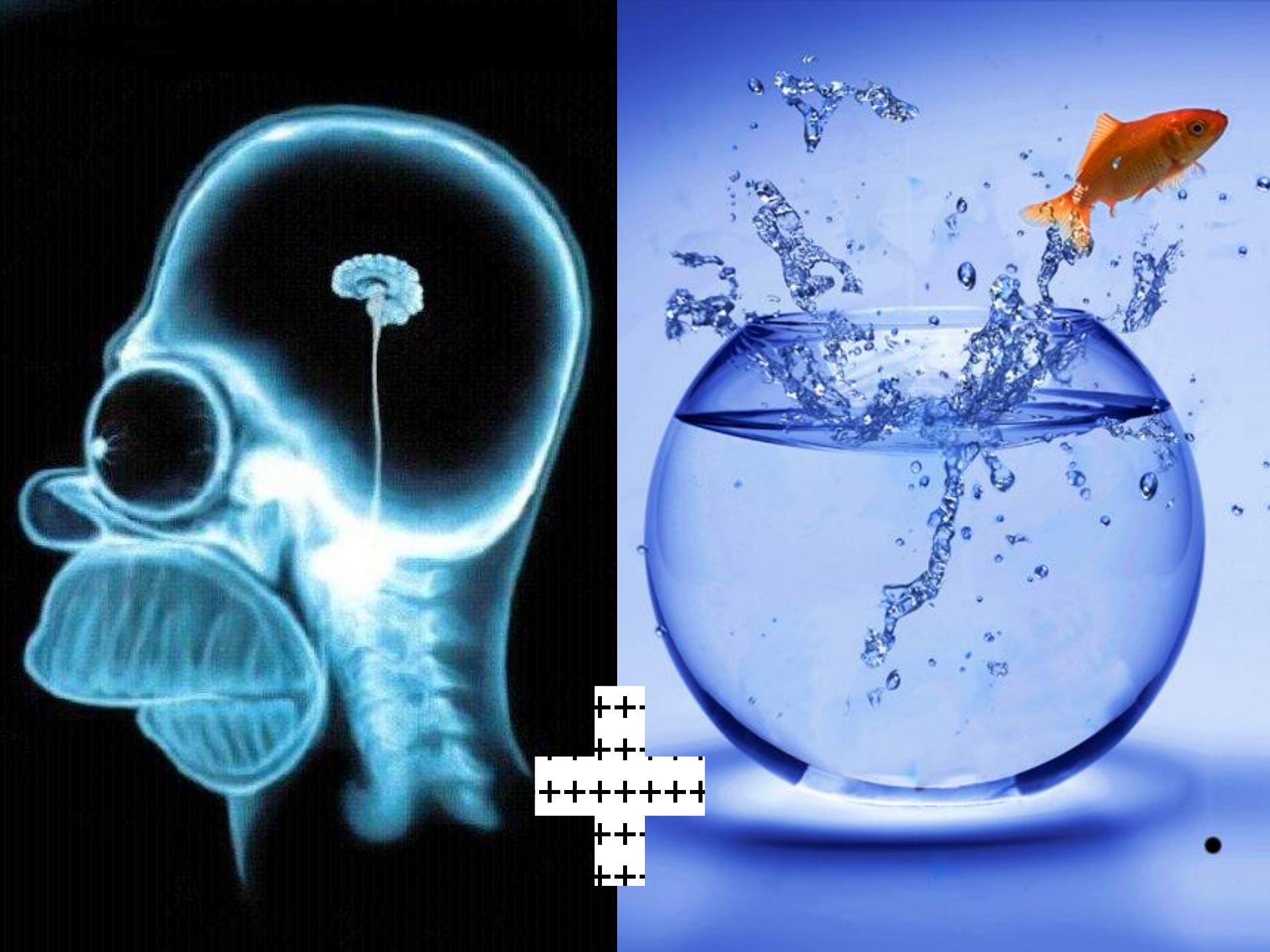
Schmutz im Code stellt eine reale Gefahr dar!

Clean Code kostet zusätzliche Zeit
in der Entwicklung!

Clean Code spart Zeit
in der Wartung!


Short Time
Cycle
ERROR


Long Time Cycle
Short Time Cycle
ERROR
Welcome to v2.7



Agenda

- Einleitung
- Ursachen von Dirty-Code
- Die Clean-Code-Developer-Bewegung
- Das Clean Code Controlling
- Meinungen zu Clean-Code
- Was ist die Take-Home-Message?

Weiterführende Literatur

- Clean Code, Robert C. Martin, Prentice Hall, 2008
- The Clean Coder, Robert C. Martin, Prentice Hall, 2011
- Clean Coder: Verhaltensregeln für professionelle Programmierer, Robert C. Martin, Addison-Wesley, 2011
- The Pragmatic Programmer, Addison-Wesley, 1999
- Soft Skills für Softwareentwickler, dpunkt-Verlag, 2010
- http://www.clean-code-developer.de
- http://de.wikipedia.org/wiki/Clean_Code
- http://www.clean-code.info

Weiterführende Literatur

- Effective Java (2nd Edition), Joshua Bloch, Addison-Wesley, 2008
- Der Weg zum Java-Profi, dpunkt-Verlag, 2011
- Bug-Patterns in Java, Eric Allen, Apress, 2002
- More Java Pitfalls, Wiley, 2003

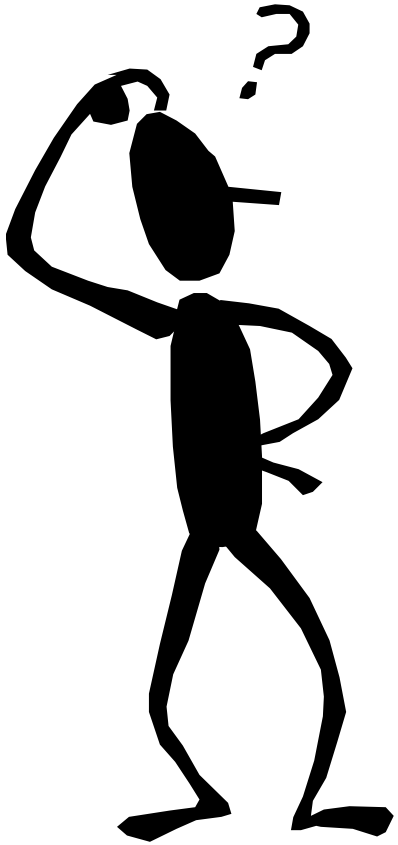
Take-Home-Message

Selbstkontrolle und Wertesystem konsequent anzuwenden, ist schwer!

Möchten wir…

- …lesbaren & langlebigen Code? ⇨ Clean-Code-Prinzipien anwenden
- …effektiv & effizient arbeiten? ⇨ Clean-Code-Praktiken anwenden
- …effektiv & effizient bleiben? ⇨ Clean-Code-Controlling anwenden

Fragen?



Diskussion

- Worin siehst Du Probleme in der Anwendung?
- Welche Art von Unterstützung brauchst Du im Programmier-Alltag?
- Ist diese Selbstkontrolle so schwer, dass man ein Clean Code Controlling benötigt?

Bildernachweise

Die in diesem Vortrag verwendeten Bilder stammen von folgenden Quellen:

Folie 6: http://www.flickr.com/photos/pringels/3139474250/sizes/l/in/photostream/

Folie 7: http://www.flickr.com/photos/23313526@N07/4948442428/sizes/l/in/photostream/
http://www.flickr.com/photos/29747502@N03/2784238062/sizes/l/in/photostream/

Folie 8: http://www.flickr.com/photos/buridansesel/6163446452/sizes/l/in/photostream/

Folie 9: http://office.microsoft.com/en-us/images
http://officeimg.vo.msecnd.net/en-us/images/MB900443111.jpg
http://officeimg.vo.msecnd.net/en-us/images/MB900443251.jpg

Folie 10: http://office.microsoft.com/en-us/images

Folie 11: http://photos.signonsandiego.com/album55/mud02

Folie 13: http://www.sxc.hu/browse.phtml?f=download&id=1099152

Folie 14: http://upload.wikimedia.org/wikipedia/commons/9/91/22_West_-_view_from_window.jpg

Bildernachweise

Folie 16: http://www.clker.com

Folie 18: http://www.sxc.hu/browse.phtml?f=download&id=866819

Folie 26: http://www.sxc.hu/browse.phtml?f=download&id=544619

Folie 27: http://commons.wikimedia.org/wiki/File:Change.jpg

Folie 30: http://www.flickr.com/photos/ionics/6338284584

Folie 31: http://www.f1online.de
http://www,flickr.com

Folie 33: http://www.clker.com

Folie 35: http://www.clker.com

Folie 36: http://www.clean-code-developer.de

Folie 38: http://photos.signonsandiego.com/album55/mud02

Bildernachweise

Folie 41, 43, 45: http://www.clean-code-developer.de

Folie 58: http://www.flickr.com/photos/oskay/437341603/

Folie 60, 61: http://www.clker.com

Folie 62: http://openclipart.org/people/DooFi/DooFi_Piggybank.png
http://openclipart.org/people/lolonene/1309969161.png

Folie 65: http://freepostermaker.com/uploads/saved_posters/free-poster-dnxeoi88fg-WILLIES-WASH.jpg

Folie 69: http://office.microsoft.com/de-de/images/results.aspx?qu=frau+computer&ex=1&ctt=1#ai:MP900430491|
http://office.microsoft.com/de-de/images/results.aspx?ex=2&qu=frau%20b%C3%BCro%20stress#ai:MP900422409|mt:2|

Folie 72: http://www.hborchert.de/medihumor.htm
http://hdfreewallpaper.info/fishy-ubuntu-1920-x-1080.html

Folie 76: http://www.clker.com

Hinweis: Sämtliche hier nicht aufgeführte Bilder (z.B. auf Folie 28) wurden von den Autoren selbst erstellt.

Gesellschaft für
Informations- und
Kommunikationssysteme mbH

www.iks-gmbh.com