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**

26.06.2012

Moderne Architekturen


Design Patterns

Der Code

Wofür steht Clean Code?

&

Software-qualität

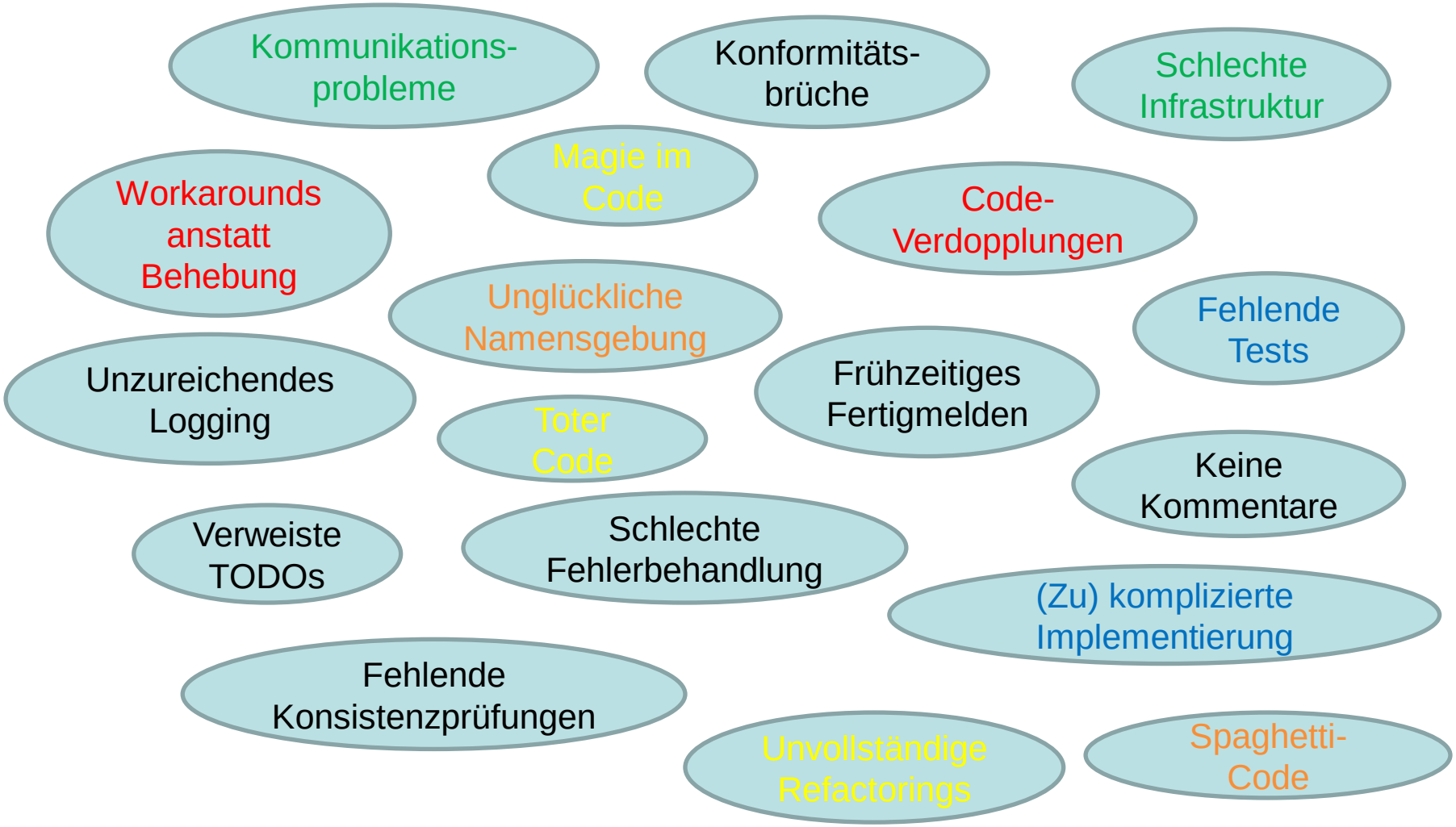
Effektive Softwareentwicklung

Agenda

- Einleitung
- **Ursachen von Dirty Code**
- Die Clean Code Developer Bewegung
- Das Team Clean Coding
- Meinungen zu Clean Code und zum Team Clean Coding


Ursachen
von
Dirty Code

Häufige Unsitten



Grund 1: Unvollständige Umsetzung

Unvollständige Umsetzung

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

Entwickler: „Ja, aber erst mal nur ein Schnellschuss, das kann nicht so bleiben…“

Kunde & Projektleiter: „…sieht doch gut aus! Mehr brauchen wir doch gar nicht!“

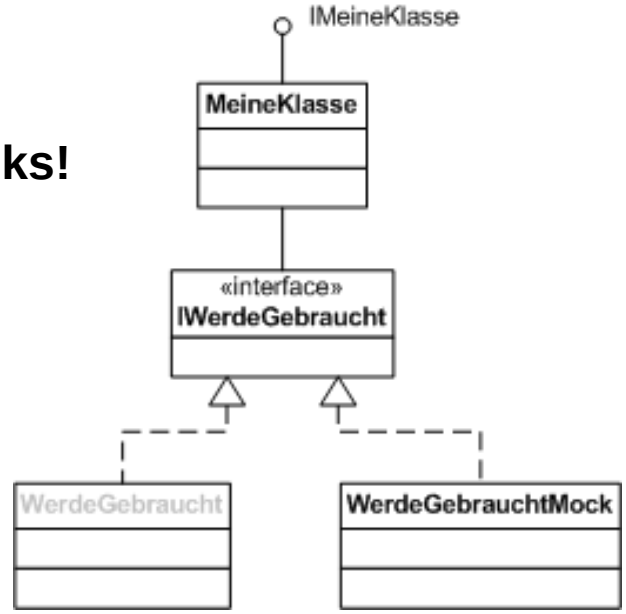
Welcher Entwickler hat da den Mut, sich durchzusetzen?

Unvollständige Umsetzung

Projektleiter A: „Stell‘ mir mal deinen aktuellen Stand zur Verfügung, Projektleiter B braucht den, damit sein Team weiterarbeiten kann!“

Stimmt nicht (mehr), denn es gibt Mocks!

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



Unvollständige Umsetzung

„Lass gut sein, das ziehen wir später gerade. Hauptsache, es läuft!“

Die Erfahrung zeigt:

Nachträgliches Refactoring funktioniert nur sehr selten, weil

…es zu mühsam ist (erneutes Hineindenken)
…das Risiko ist groß ist, Fehler einzubauen
…größere Maßnahmen zeitlich kaum abschätzbar sind
…es (deswegen) Widerstand bei der Projektleitung gibt

Unvollständige Umsetzung

Entwickler: „Fertig, es läuft!“

Projektleiter: „Prima, dann können wir endlich liefern!“

Definition of Done

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)

Grund 2:
Mangelnde Motivation

Mangelnde Motivation: Broken Window

Entwickler: „Ich hab damit nichts zu tun, das ist nicht von mir!“

„Darauf kommt es jetzt auch nicht mehr an!“

„Warum soll ich anfangen, hier sauber zu implementieren?“

Mangelnde Motivation: Die x-te Änderung

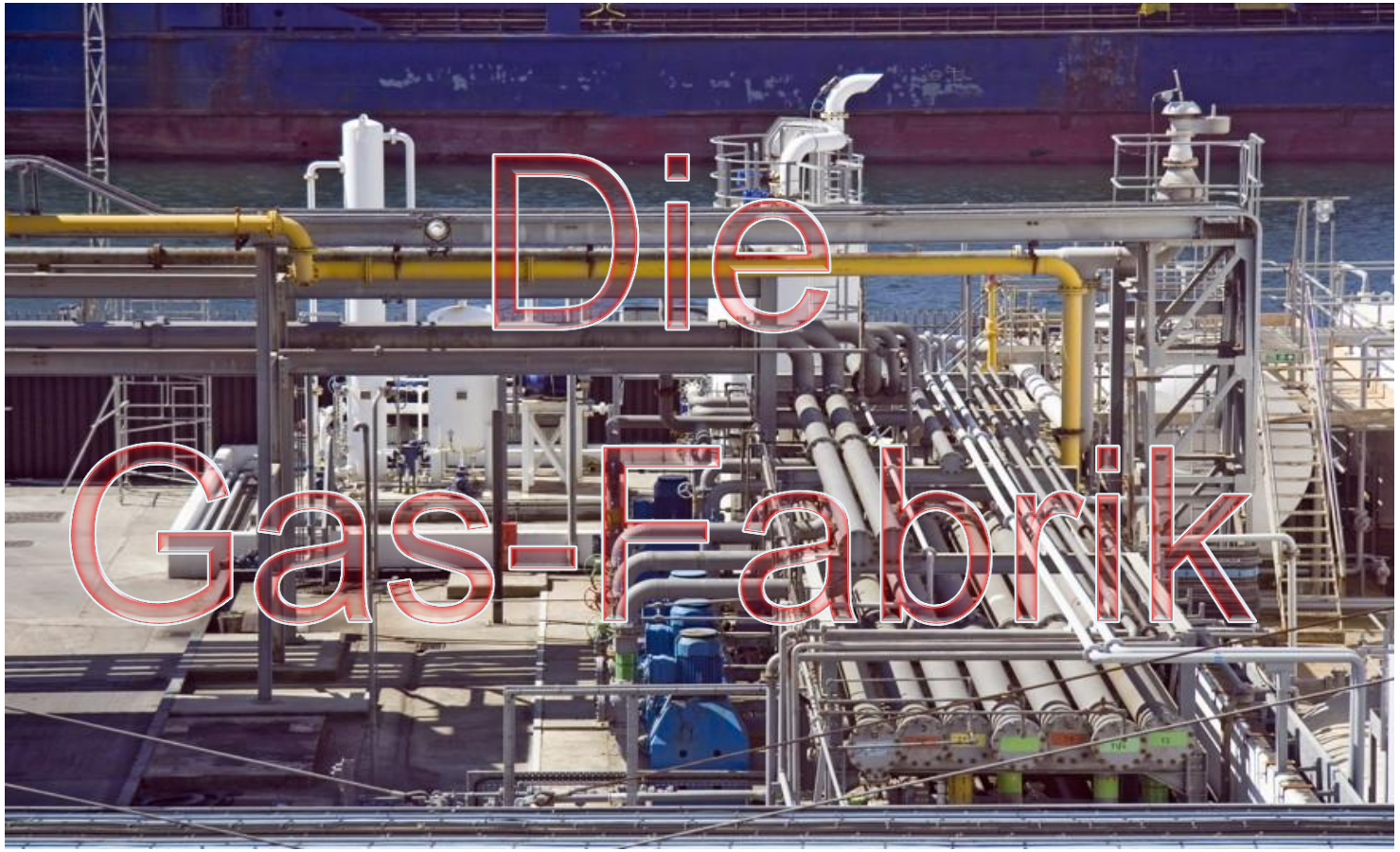
Entwickler: „Das war so gar nicht spezifiziert!“

„Wie oft muss ich da noch ran?“

Grund 3:
Unlogische Realisierung

Unlogische Realisierung: Die Gasfabrik

Entwickler: „Das muss auch noch rein, denn so wird das richtig gut!“



Unlogische Realisierung: Konformitätsbrüche

„Ich mach das aber anders als die anderen!“

„Was interessiert mich, was die anderen gemacht haben?“

Grund 4:
Schlechte Rahmenbedingungen

Schlechte Rahmenbedingungen: Menschliche Defizite

Fortbildungsdefizite

– Mangelndes Interesse

Kommunikationsprobleme

– Sender-Empfänger-Probleme
– Schlechtes Team-Klima

Motivationsprobleme

– Kein Bewertungssystem für Lob und Tadel

Schlechte Rahmenbedingungen: Organisatorische Defizite

Unzureichende Weiterbildungsmöglichkeiten

Unklare Rollen,

Unklare Prozesse,

Unklare Richtlinien

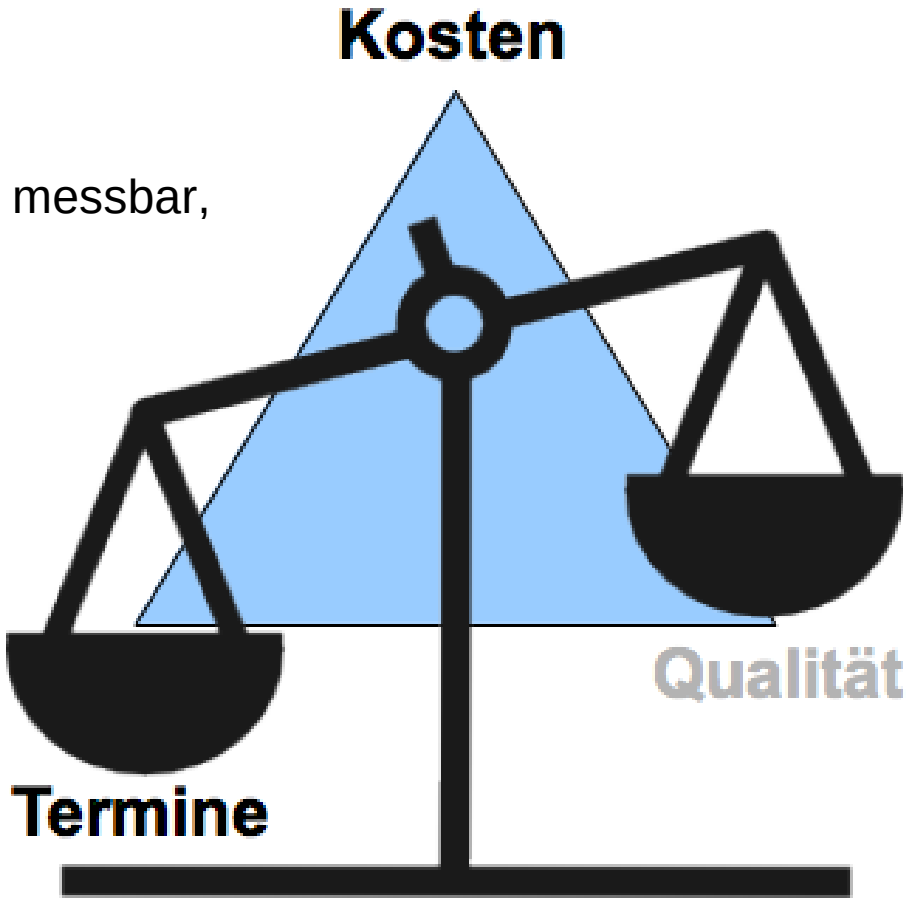
…

Grund 5:
Termine

Termine

Dilemma:

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



Ursachen für Dirty Code

- **Unvollständige Umsetzung**
- **Mangelnde Motivation**
- **Unlogische Realisierung**
- **Schlechte Rahmenbedingungen**
- **Termine**

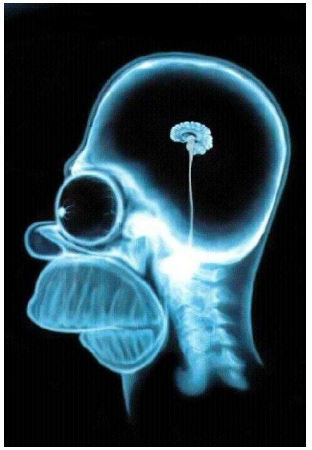
Moment mal !!!
Sind wir etwa faul, feige und unfähig???
Nein! Aber wir können besser werden!

Agenda

- Einleitung
- Ursachen von Dirty Code
- **Die Clean Code Developer Bewegung**
- Das Team Clean Coding
- Meinungen zu Clean-Code

Was heißt clean?

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



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.


Mr. Clean
Die Clean Code Developer

Kernaussagen der Clean-Code-Developer-Bewegung

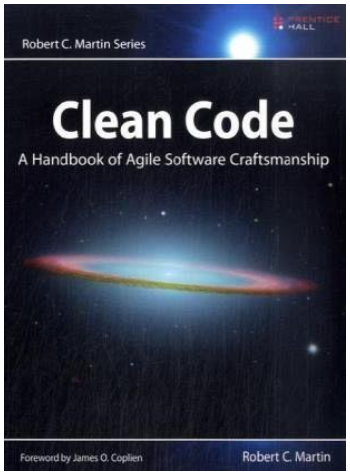
- Disziplin („Professionalität“)
  Ständige Selbstkontrolle („Bewusstsein“) durch konsequente Anwendung des inneren Wertesystems („Prinzipien“)
- Wertesystem
  Das Buch „Clean Code“ ist wert, als allgemeingültiges Wertesystem anerkannt zu werden.
- Programmieralltag
  Das CCD-Grade-System hilft, das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

Die Clean Code Grade:

	Schwarz	Rot	Orange	Gelb	Grün	Blau	Weiss
Prinzipien		● Don´t Repeat Yourself (DRY), ● Keep it simple, stupid (KISS) ● Vorsicht vor Optimierungen (VvO) ● Favour Composition over Inheritance (FCoI)	● Single Level of Abstraction (SLA) ● Single Responsibility Principle (SRP) ● Separation of Concerns (SoC) ● Source Code Konventionen	● Interface Segregation Principle ● Dependency Inversion Principle ● Liskov Substitution Principle ● Principle of Least Astonishment ● Information Hiding Principle	● Open Closed Principle ● Tell, don´t ask ● Law of Demeter	● Entwurf und Implementation überlappen nicht ● Implementa-tion spiegelt Entwurf ● You Ain´t Gonna Need It (YAGNI)	
Praktiken		● Die Pfadfinderregel beachten ● Root Cause Analysis ● Ein Versionskon-trollsystem einsetzen ● Einfache Refakto-risierungsmuster an-wenden (ER) ● Täglich reflektieren	● Issue Tracking ● Automatisierte Integrationstests ● Lesen, Lesen, Lesen (LLL) ● Reviews	● Automatisierte Unit Tests ● Mockups (Testattrappen) ● Code Coverage Analyse ● Teilnahme an Fachveranstal-tungen ● Komplexe Re-faktorisierungen	● Continuous Integration I ● Statische Codeanalyse (Metriken) ● Inversion of Control Container ● Erfahrung weitergeben ● Messen von Fehlern	● Continuous Integration II ● Iterative Entwicklung ● Komponenten-orientierung ● Test first	

Was heißt Clean Code noch?

Def. 4: Clean Code ist ein Buch mit einem umfassenden Regelwerk, das als Sprachen über-greifendes Wertesystem (Best Practices) gelten kann.



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

Was bewirkt Clean Code?

Code

Leicht verständlich

Einfach änderbar

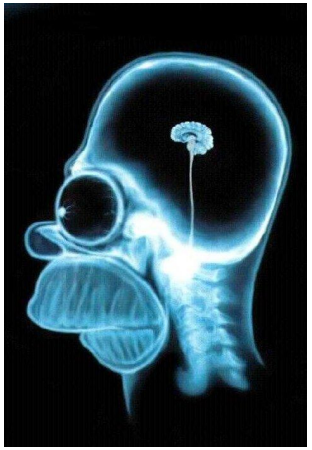
Gut testbar

flexibel

Entwicklung

wartbar

effektiv



Agenda

- Einleitung
- Ursachen von Dirty Code
- Die Clean Code Developer Bewegung
- **Das Team Clean Coding**
- Meinungen zu Clean Code und zum Team Clean Coding


Mr. Clean
Mr. Clean
Mr. Clean
Mr. Clean
Mr. Clean
Das Team
Clean Coding
Mr. Clean
Mr. Clean
Mr. Clean
Mr. Clean
Mr. Clean

Das Team Clean Coding

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 Team Clean Coding

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 Team Clean Coding

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 Team Clean Coding (Legislative)

Was muß entschieden werden?

- Was bedeutet für uns die Definition of Done?
- Wie gehen wir mit unfertigem C

Code Tagging:

FIXMEs, TODOs und REFACTORE_MEs:

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

DONE ⇨ FIXMEs raus, TODOs raus oder in den Issue-Tracker

Das Team Clean Coding

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 Team Clean Coding

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 Team Clean Coding(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.

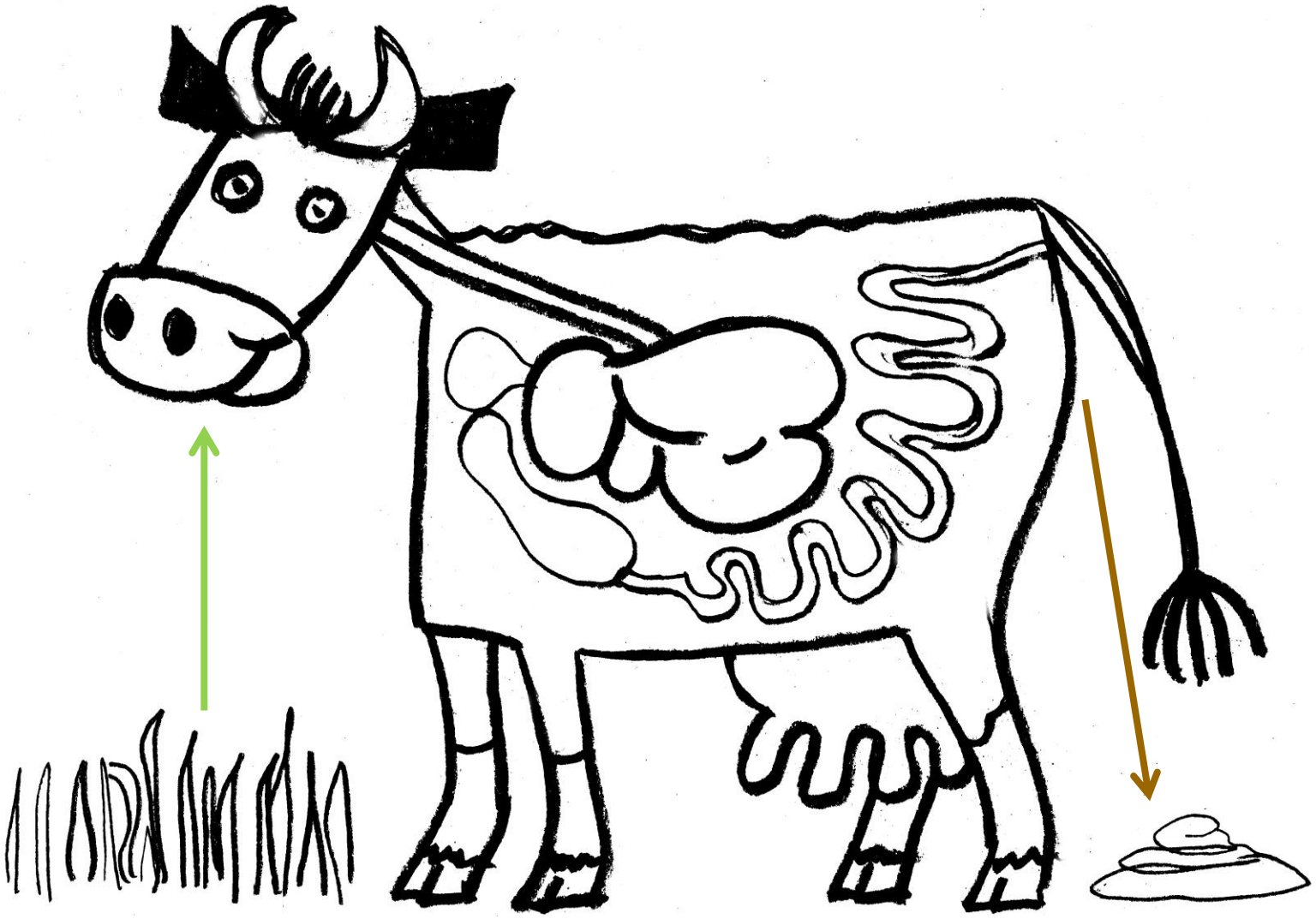
Was hilft wann?

Dirty Code Ursachen	Clean Code Dev.	Team Clean Coding
Unklare DOD		DOD definieren
Hauptsache, es läuft!	Test first	DOD anwenden, 4iReviews
Gasfabrik	KISS, YAGNI, VvO	Team-Reviews, 4iReviews
X-te Änderung		Code Tagging
Broken Windows	Pfadfinderregel	Code Tagging
Konformitätsbrüche	Reviews, ER, SCC	4iReviews
Fortbildung	LLL, TAF	Team-Reviews, Pairprogramming
Kommunikation	Erfahrung weitergeben	Wege zur Einigung im Team
Organisation		Rollen & Prozesse definieren
Motivation	Bewusstsein	Anerkennung, 1€-Regel
Termine	Mentale Einstellung	DOD anwenden

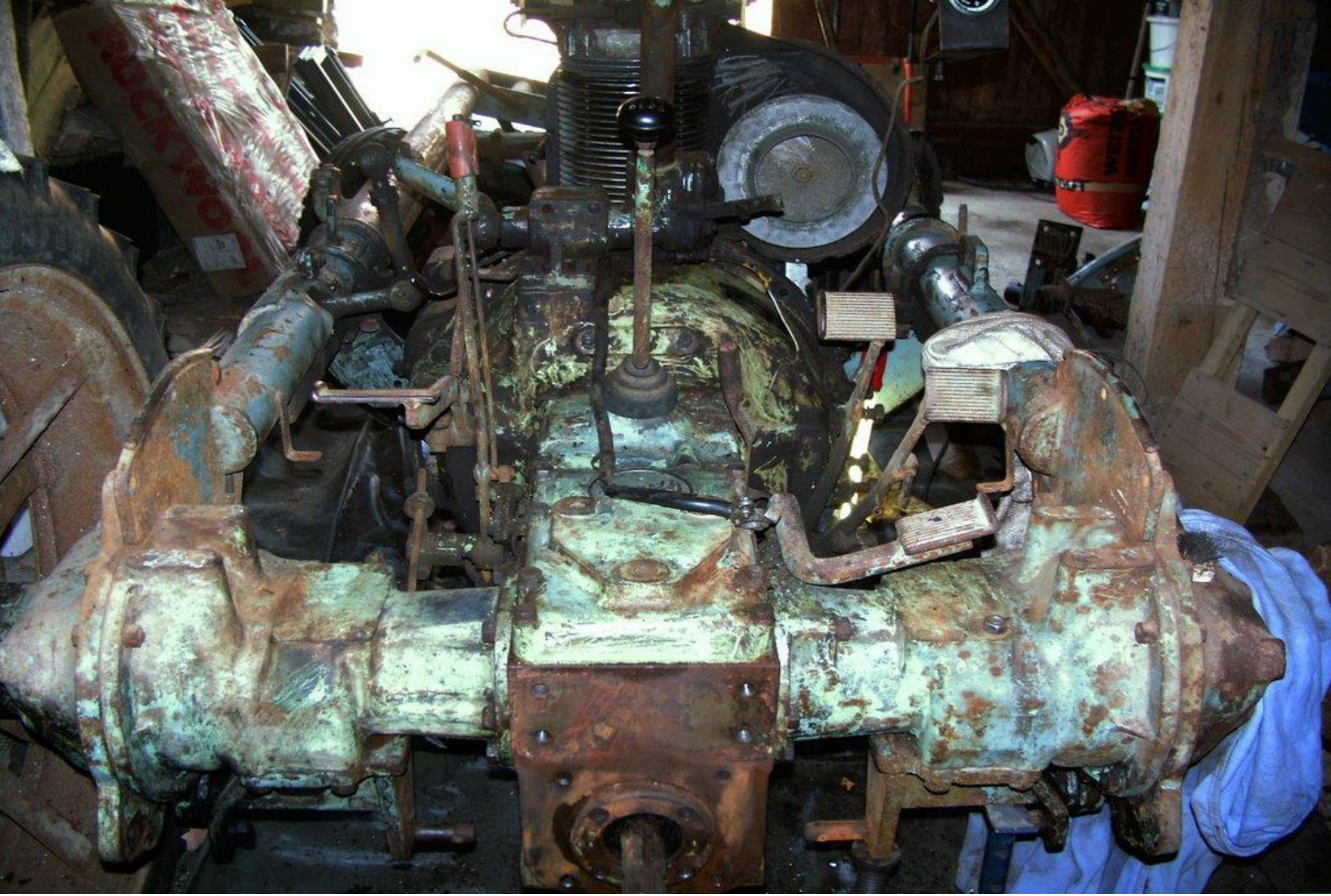
Agenda

- Einleitung
- Ursachen von Dirty Code
- Die Clean Code Developer Bewegung
- Das Team Clean Coding
- **Meinungen zu Clean Code und zum Team Clean Coding**

Clean Code ist gut, aber manchmal reicht das nicht aus. Ästhetik!



Ein Organismus verträgt nur ein begrenztes Maß an Schadstoffen.



Wichtig sind Funktionsfähigkeit, Stabilität, Wartbarkeit und Erweiterbarkeit.
Clean-Code ist was für Spießer – wichtig ist nur, dass es läuft!
Dazu braucht man kein Clean Code!

Sauberer Code - Hohe Testabdeckung

Produktionscode

Testabdeckung

Testcode

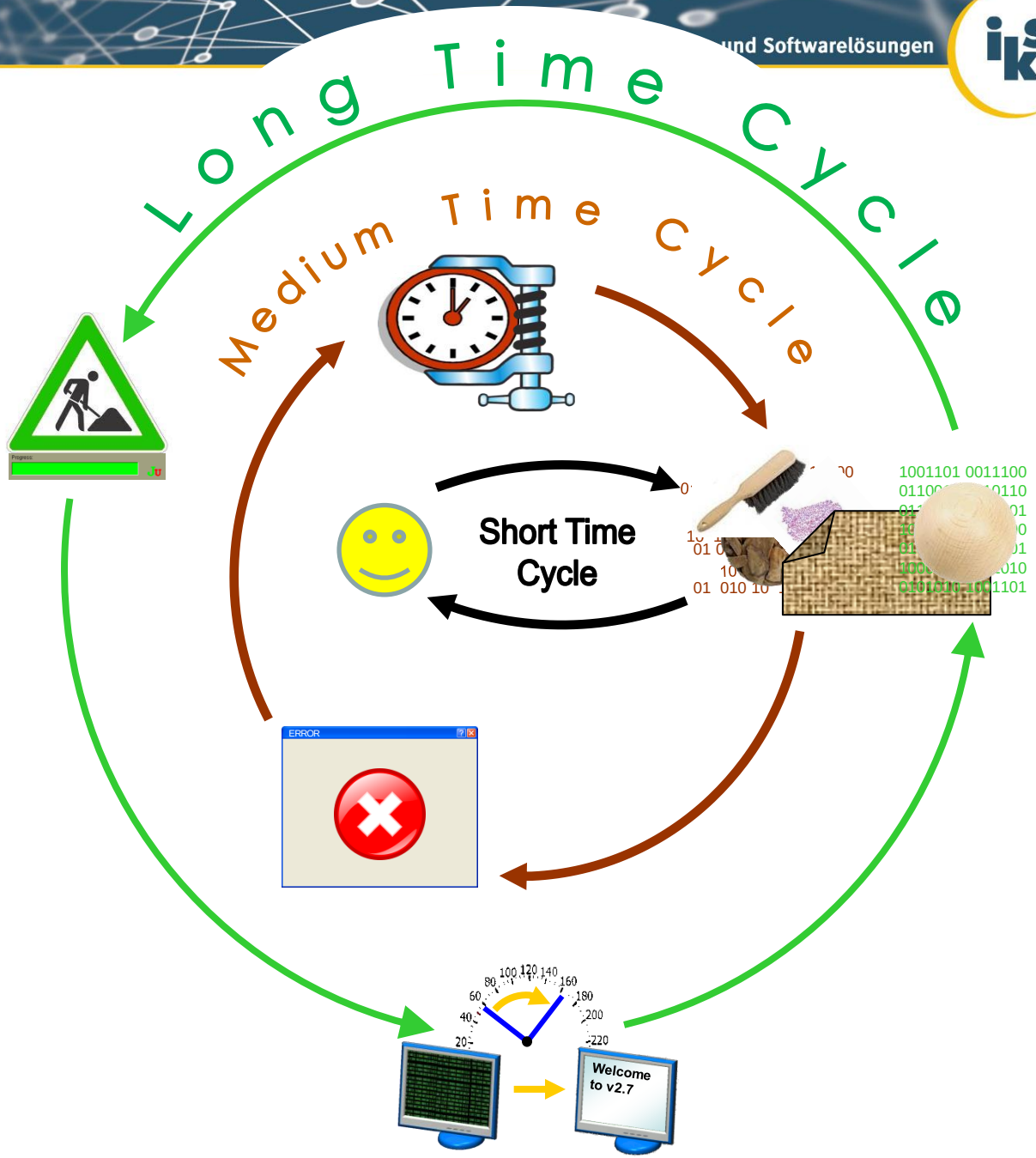
Schmutz im Code stellt eine reale Gefahr dar!

Clean Code kostet zusätzliche Zeit
in der Entwicklung!

Clean Code spart Zeit
in der Wartung!

Gesellschaft für
Informations- und
Kommunikationssysteme mbH

Wie wirkt sich die Anwendung der Best practices aus?



Wofür steht Clean Code?

&

Software-qualität

Effektive Softwareentwicklung

Das Produkt
(finales Ziel mit Selbstzweck)

Der Entwicklungs-prozess
(Zwischenziel, nur Mittel zum Zweck)

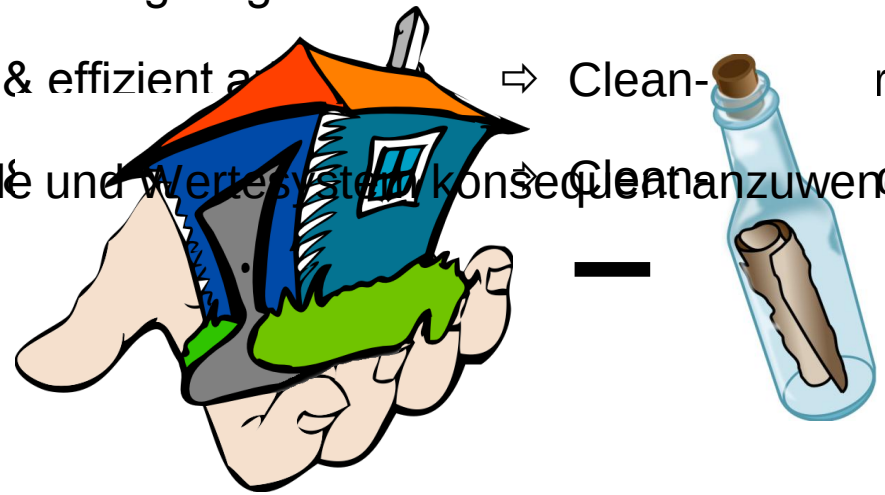
Nicht-funktionale Qualitätsmerkmale von Software	
korrekt, skalierbar, performant, stabil, effizient, sicher (Security), zuverlässig, gesetzeskonform, gut bedienbar	Externe Qualität
wirtschaftlich, testbar, erweiterbar, veränderbar, analysierbar, wartbar	Interne Qualität

Take-Home-Message

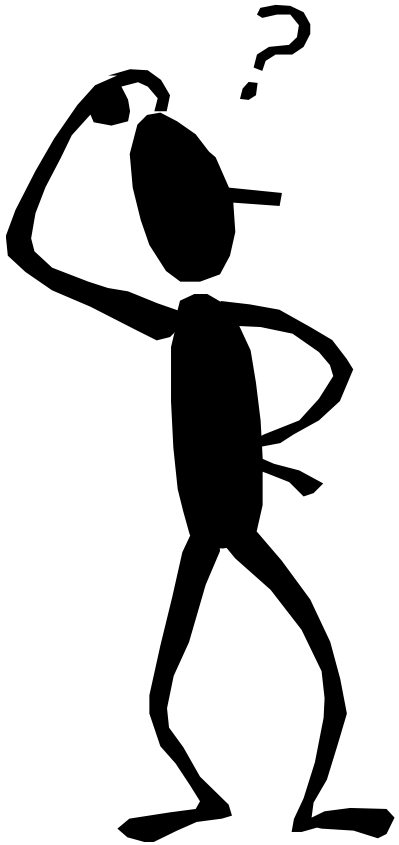
Möchten wir…

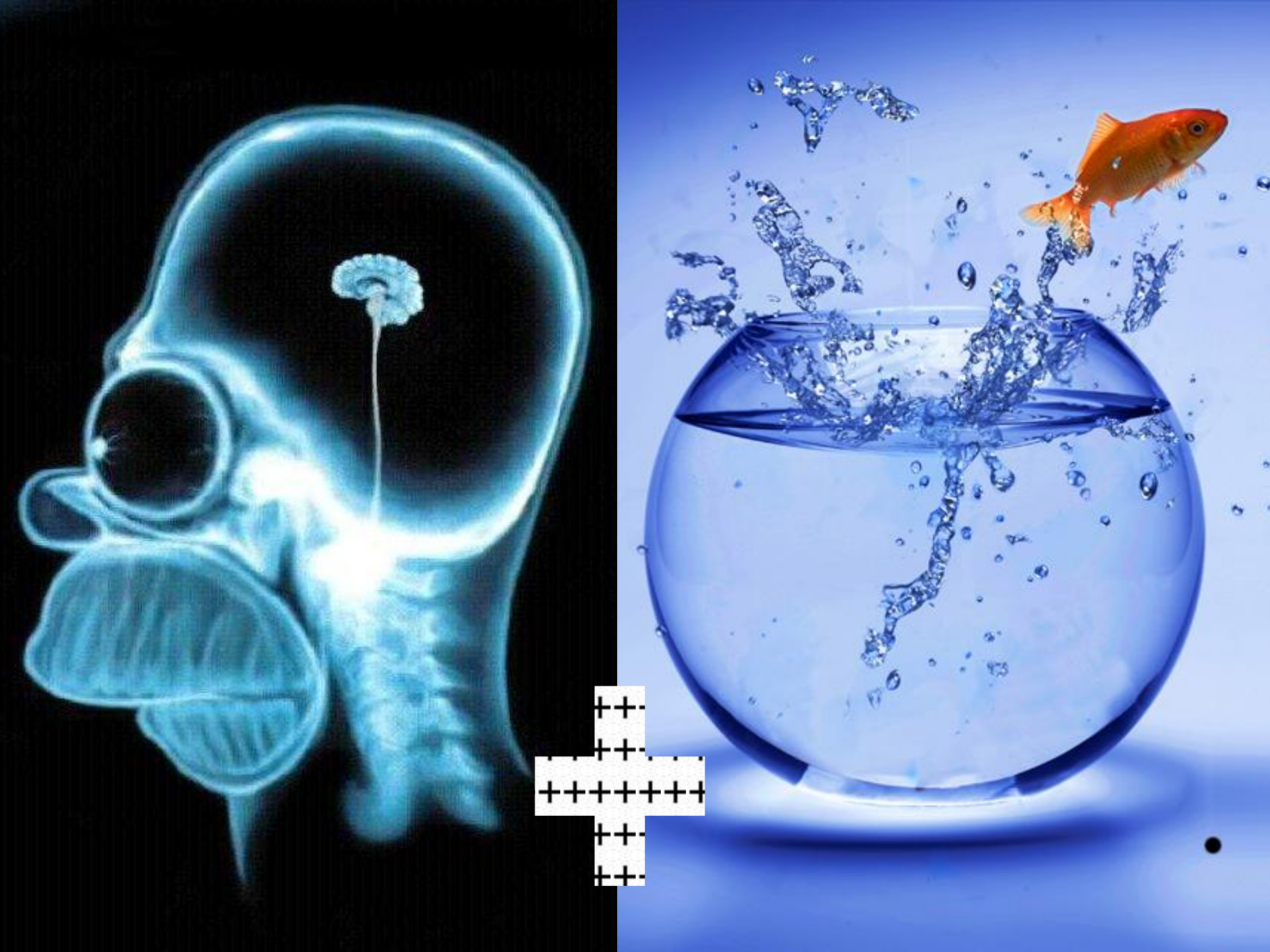
- …lesbaren & langlebigen Code? ⇨ Clean-Code-Prinzipien anwenden
- …effektiv & effizient a… ⇨ Clean-…raktiken anwenden
- …effektiv & … ⇨ Clean-…ontrolling anwenden

Selbstkontrolle und Wertesystem konsequent anzuwenden, ist schwer!



Fragen?





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

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

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 Team Clean Codingbenötigt?

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

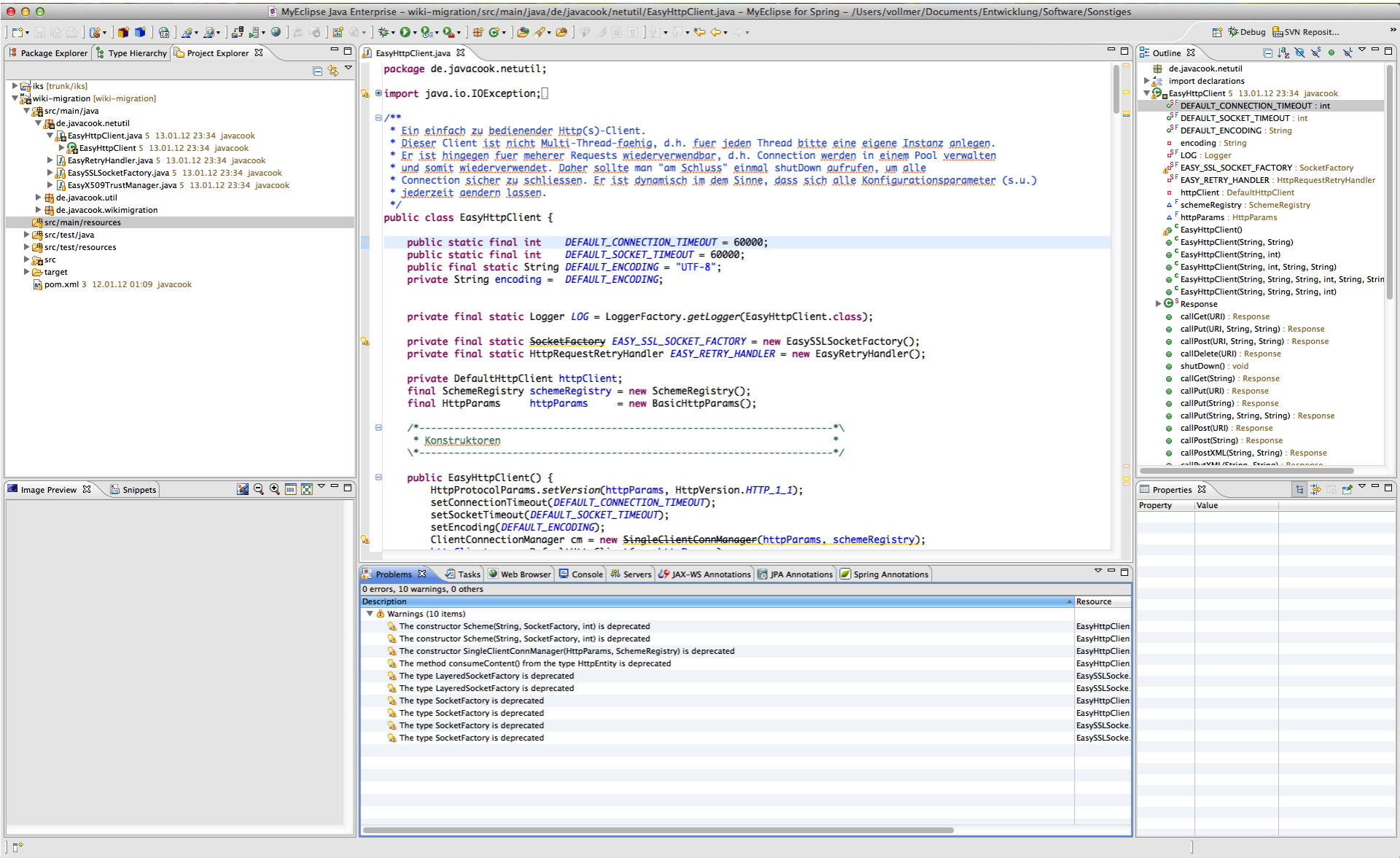
Folie 69: http://office.microsoft.com/de-de/images/results.aspx?qu=frau+computer&ex=1&ctt=1#ai:MP900430491|
http://office.microsoft.com/de-de/images/results.aspx?ex=2&qu=frau%20b%C3%BCro%20stress#ai:MP900422409|mt:2|

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

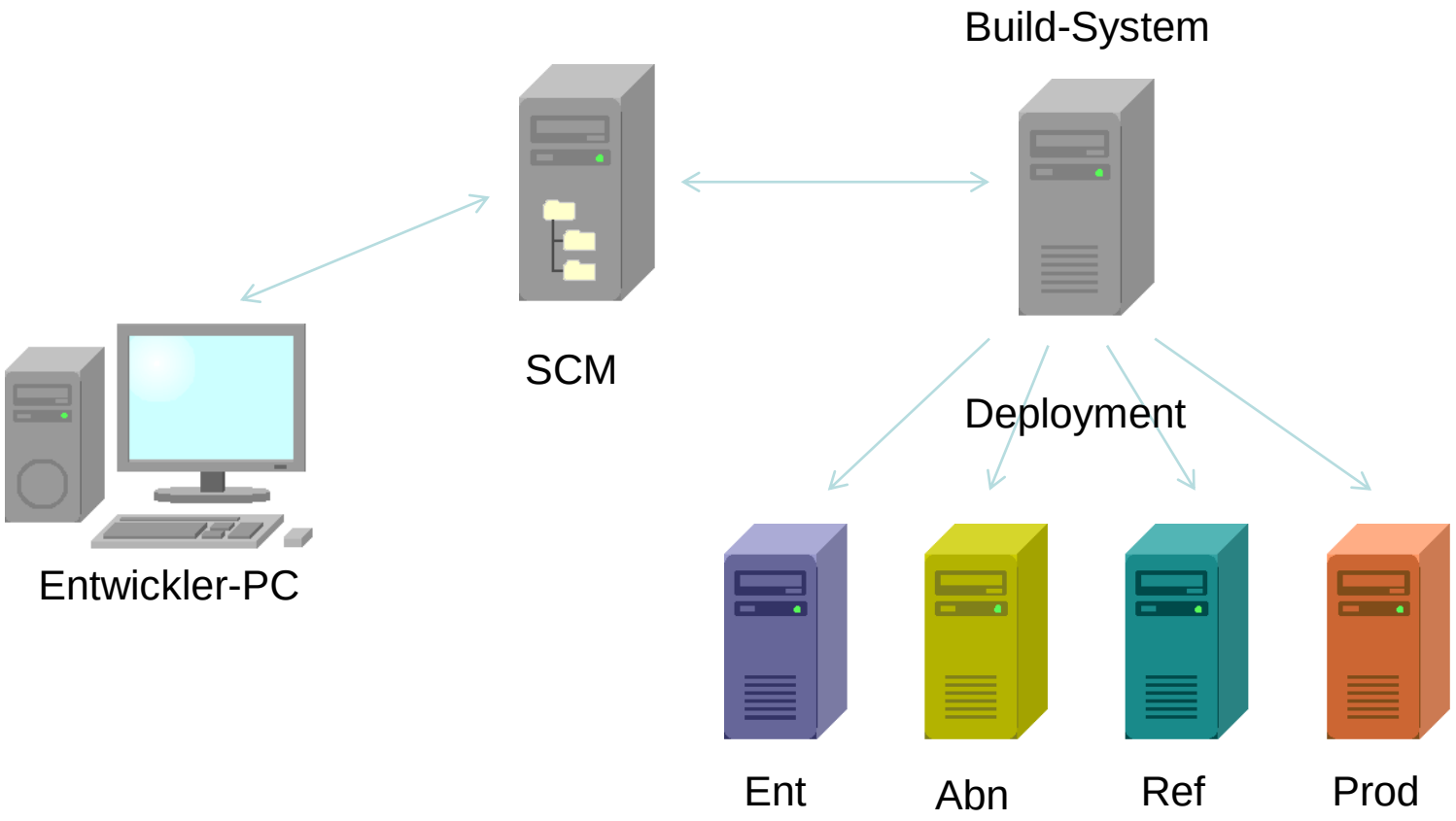
Folie 76: http://www.clker.com

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

Eclipse-IDE


MyEclipse Java Enterprise – wiki-migration/src/main/java/de/javacook/netutil/EasyHttpClient.java – MyEclipse for Spring – /Users/vollmer/Documents/Entwicklung/Software/Sonstiges
Package Explorer
Type Hierarchy
Project Explorer
iks [trunk/iks]
wiki-migration [wiki-migration]
src/main/java
de.javacook.netutil
EasyHttpClient.java 5 13.01.12 23:34 javacook
EasyHttpClient 5 13.01.12 23:34 javacook
EasyRetryHandler.java 5 13.01.12 23:34 javacook
EasySSLSocketFactory.java 5 13.01.12 23:34 javacook
EasyX509TrustManager.java 5 13.01.12 23:34 javacook
de.javacook.util
de.javacook.wikimigration
src/main/resources
src/test/java
src/test/resources
src
target
pom.xml 3 12.01.12 01:09 javacook
Image Preview
Snippets
EasyHttpClient.java
package de.javacook.netutil;
import java.io.IOException;
/**
* Ein einfach zu bedienender Http(s)-Client.
* Dieser Client ist nicht Multi-Thread-faehig, d.h. fuer jeden Thread bitte eine eigene Instanz anlegen.
* Er ist hingegen fuer meherer Requests wiederverwendbar, d.h. Connection werden in einem Pool verwalten
* und somit wiederverwendet. Daher sollte man "am Schluss" einmal shutDown aufrufen, um alle
* Connection sicher zu schliessen. Er ist dynamisch im dem Sinne, dass sich alle Konfigurationsparameter (s.u.)
* jederzeit aendern lassen.
*/
public class EasyHttpClient {
public static final int DEFAULT_CONNECTION_TIMEOUT = 60000;
public static final int DEFAULT_SOCKET_TIMEOUT = 60000;
public final static String DEFAULT_ENCODING = "UTF-8";
private String encoding = DEFAULT_ENCODING;
private final static Logger LOG = LoggerFactory.getLogger(EasyHttpClient.class);
private final static SocketFactory EASY_SSL_SOCKET_FACTORY = new EasySSLSocketFactory();
private final static HttpRequestRetryHandler EASY_RETRY_HANDLER = new EasyRetryHandler();
private DefaultHttpClient httpClient;
final SchemeRegistry schemeRegistry = new SchemeRegistry();
final HttpParams httpParams = new BasicHttpParams();
/*---*\
* Konstruktoren *
---/
public EasyHttpClient() {
HttpProtocolParams.setVersion(httpParams, HttpVersion.HTTP_1_1);
setConnectionTimeout(DEFAULT_CONNECTION_TIMEOUT);
setSocketTimeout(DEFAULT_SOCKET_TIMEOUT);
setEncoding(DEFAULT_ENCODING);
ClientConnectionManager cm = new SingleClientConnManager(httpParams, schemeRegistry);
Outline
de.javacook.netutil
import declarations
EasyHttpClient 5 13.01.12 23:34 javacook
DEFAULT_CONNECTION_TIMEOUT : int
DEFAULT_SOCKET_TIMEOUT : int
DEFAULT_ENCODING : String
encoding : String
LOG : Logger
EASY_SSL_SOCKET_FACTORY : SocketFactory
EASY_RETRY_HANDLER : HttpRequestRetryHandler
httpClient : DefaultHttpClient
schemeRegistry : SchemeRegistry
httpParams : HttpParams
EasyHttpClient()
EasyHttpClient(String, String)
EasyHttpClient(String, int)
EasyHttpClient(String, int, String, String)
EasyHttpClient(String, String, String, int, String, Strin
EasyHttpClient(String, String, String, int)
Response
callGet(URI) : Response
callPut(URI, String, String) : Response
callPost(URI, String, String) : Response
callDelete(URI) : Response
shutDown() : void
callGet(String) : Response
callPut(URI) : Response
callPut(String) : Response
callPut(String, String, String) : Response
callPost(URI) : Response
callPost(String) : Response
callPostXML(String, String) : Response
Properties
Property
Value
Problems
Tasks
Web Browser
Console
Servers
JAX-WS Annotations
JPA Annotations
Spring Annotations
0 errors, 10 warnings, 0 others
Description
Resource
Warnings (10 items)
The constructor Scheme(String, SocketFactory, int) is deprecated
The constructor Scheme(String, SocketFactory, int) is deprecated
The constructor SingleClientConnManager(HttpParams, SchemeRegistry) is deprecated
The method consumeContent() from the type HttpEntity is deprecated
The type LayeredSocketFactory is deprecated
The type LayeredSocketFactory is deprecated
The type SocketFactory is deprecated
The type SocketFactory is deprecated
The type SocketFactory is deprecated
The type SocketFactory is deprecated
EasyHttpClien
EasyHttpClien
EasyHttpClien
EasyHttpClien
EasySSLSocke.
EasySSLSocke.
EasyHttpClien
EasyHttpClien
EasySSLSocke.
EasySSLSocke.

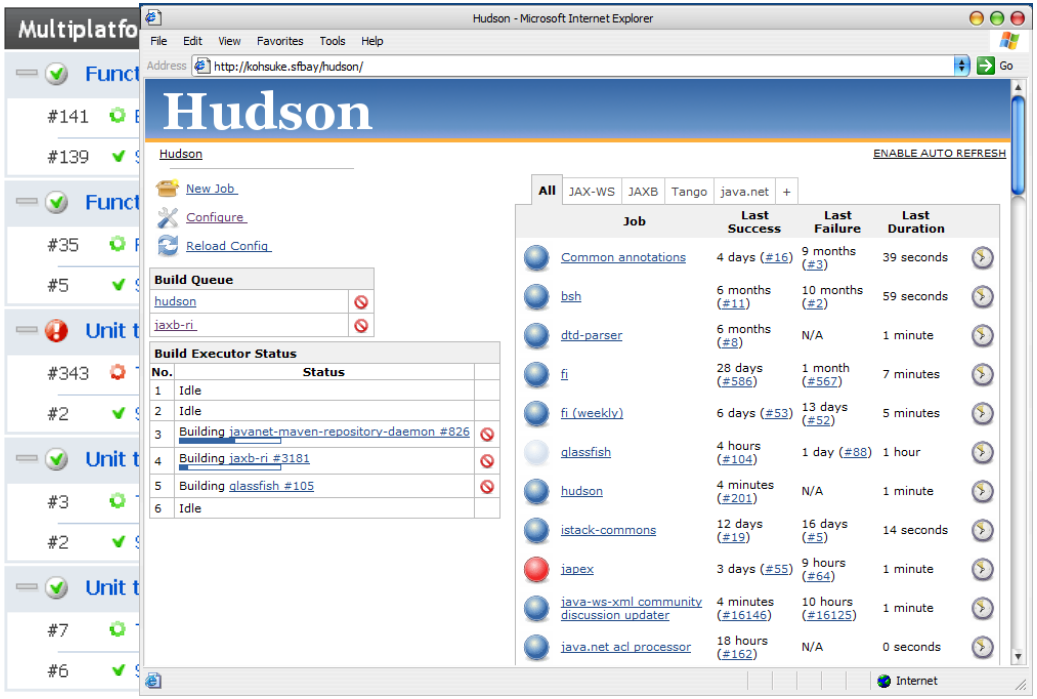
Infrastruktur Entwicklung



Continuous Build / Integration

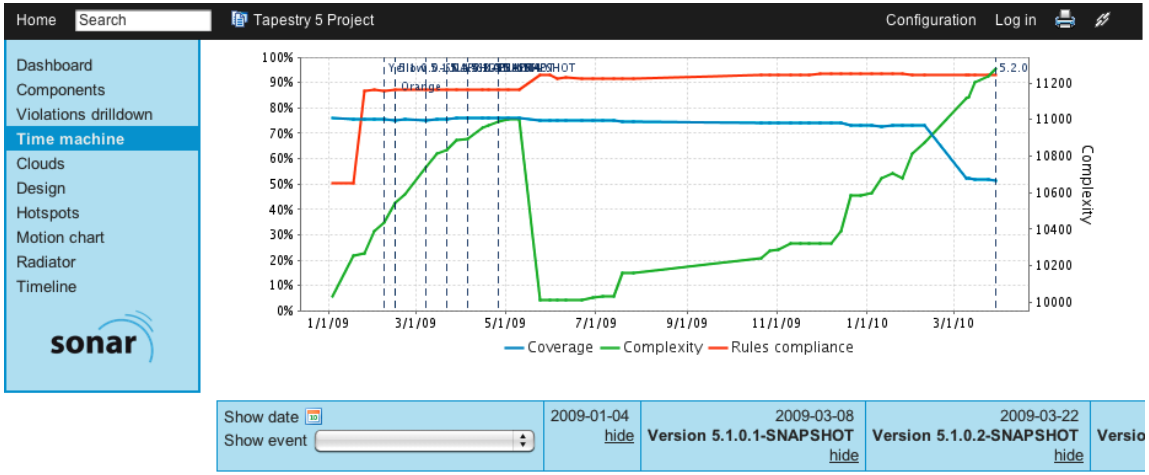
1. Diese Tools ermöglichen einen automatischen Build der eingecheckten Sourcen
2. Getriggert: Zeitgesteuert, per Hand, Einchecken
3. ANT / Maven / Skript-Integration
4. Automatisches Deployment möglich

- **TeamCity**
  - Ab einer Größe kostenpflichtig
- **Hudson, Jenkins**
  - Open Source



Qualitätssicherung

- **Sonar**
  - Open Source
  - Zusätzliches Tool zum Continuous Build
  - Berechnet zahlreiche Code-Metriken
  - Erzeugt schöne Grafiken
  - (Eigene) Plugins möglich



```
t.addCell (new Phrase(formatDateString(
sd.date == null || "".equals (sd.date) ?
sh.date.substring (6, 7) : sd.date), f));
```

Clean Code
caps
Cool Code

```
Date datum = strukturDaten.date;
if (datum == null) {
   datum = strukturHeader.date;
}
String datumFormatiert = formatDate(datum);
Phrase phrase = new Phrase(datumFormatiert, format);
table.addCell(phrase);
```

Gesellschaft für
Informations- und
Kommunikationssysteme mbH

www.iks-gmbh.com