Über Faulheit, Feigheit, Unfähigkeit **und Clean Code**

von
Jörg Vollmer
und Reik Oberrath


Das ist nicht von mir!
Faulheit
Das ist historisch gewachsen!
Warum soll gerade ich hier anfangen sauber zu machen?


Sieht schlimmer aus, als es ist...
Unfähigkeit
Kein Problem. Alles im Griff!

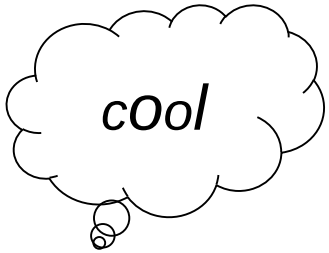
Oh nein,
wie soll das gehn…

Symptome der Grundübel

Wie sieht das konkret aus?

Symptome der Grundübel

Auszug aus einer Implementierung:




```
String[] elems = ...
...
PriorityQueue<String> pq = new PriorityQueue<>();
for (String e : elems) pq.add(e);
for (int i = 0; i < elems.length; i++) elems[i] = pq.remove();

return elems;
```

Symptome der Grundübel

Extract Method:

```
sortWithHeapSort(elems);
return elems;

void sortWithHeapSort(elems)
{
        PriorityQueue<String> pq = new PriorityQueue<>();
        for (String e : elems) pq.add(e);
        for (int i = 0; i < elems.length; i++) elems[i] = pq.remove();
}
```

Symptome der Grundübel

Unzureichende Information:

Oracle:

SQL-Fehler: ORA-00942: table or view does not exist

SQL-Fehler: ORA-00955: name is already used by an existing object

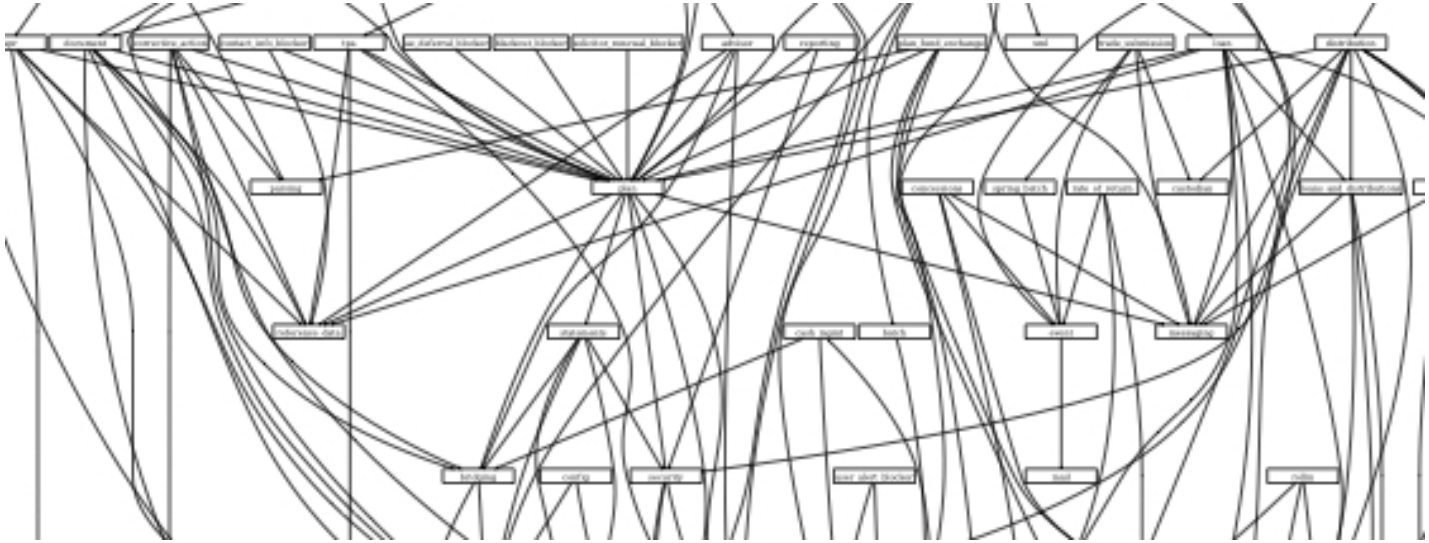
javax.ws.rs.WebApplicationException

at com.sun.jersey.server.impl.uri.rules.TerminatingRule...

java.lang.RuntimeException: null

Symptome der Grundübel

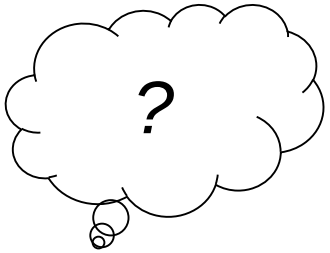
Auszug aus einem Dependency-Graphen



Quelle: http://www.samoht.com/weblog/gemcast.rb

Symptome der Grundübel

Auszug aus einer Spezifikation:



In einer Übersicht über alle noch nicht zugeordneten Geschäftspartner werden alle Geschäftspartner aller Mandanten ungleich Mandant 0 angezeigt, die bisher keinem Geschäftspartner im Mandanten 0 zugeordnet wurden. …


Unverständliches strengt an


Mr. Clean

Was bisher erreicht wurde

1. Robert C. Martin: "Clean-Code" und "The Clean Coder“
2. Martin Fowler: Refactoring
3. Kent Beck: XP
4. Agile Manifest und Agile Methoden (Scrum, Kanban,…)
5. Die deutsche *Clean-Code-Developer*-Bewegung (CCD)
6. Software Craftsmanship Community (DE: Softwerkskammer)
7. DevOps-Bewegung
8. Vorträge und Schulungen über Clean-Code

u.v.m.

Aber: Die Code-Qualität hat sich im Großen und Ganzen nicht verändert!

Publikumsfragen

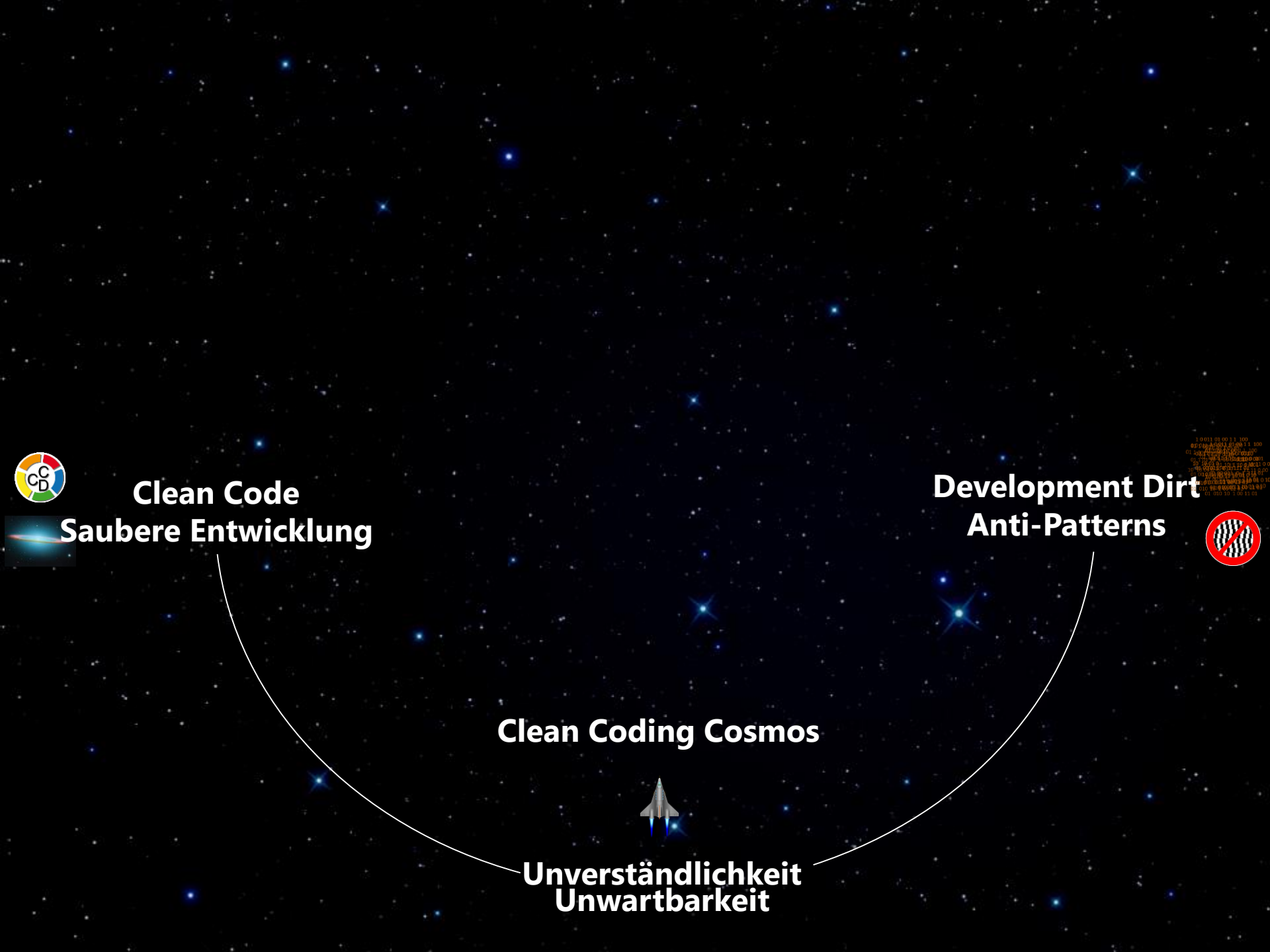
Wer hat das Buch "Clean Code" gelesen?

Wer hat sich schon mal mit den *Clean-Code-Developer*-Regeln beschäftigt?

Wer fand das falsch?

Wer wendet das bewusst und gezielt an?

Warum ist das so?


Clean Code
Saubere Entwicklung
Development Dirt
Anti-Patterns
Clean Coding Cosmos
Unverständlichkeit
Unwartbarkeit

Unser Ausgangspunkt: Was ist Clean?

Clean = effizient = evolvierbar = wartbar
= **verständlich & änderbar & testbar** (Quellcode)

Vereinfacht: Clean ist alles, was man relativ leicht versteht.
Dirty ist das, was anstrengt und aufhält.

Was ist *Clean* **nicht**?

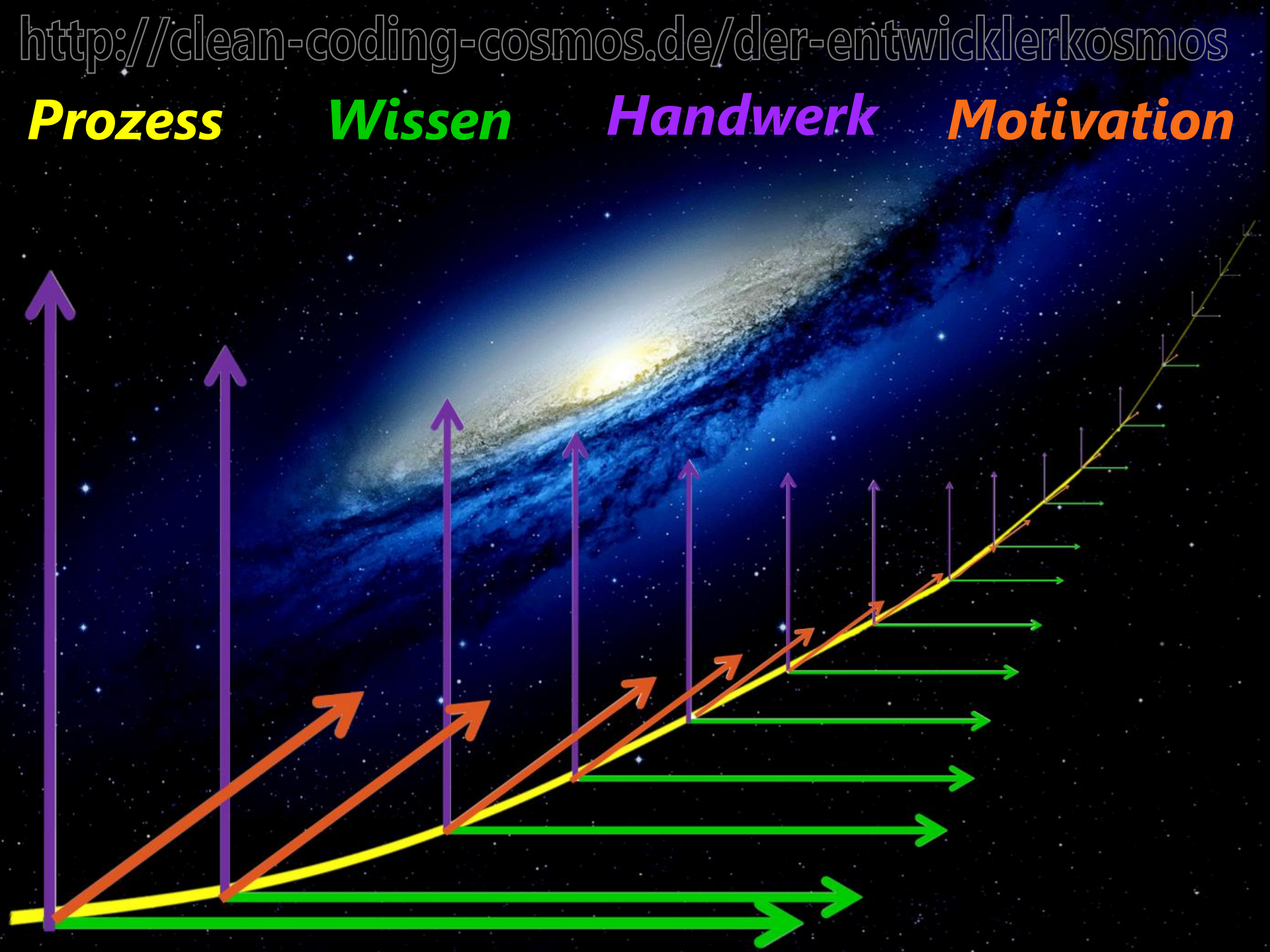
Clean ≠ penibel / akkurat

Clean ≠ korrekt

Clean ≠ elegant / cool / raffiniert

Clean ≠ Sonar einsetzen

http://clean-coding-cosmos.de/der-entwicklerkosmos



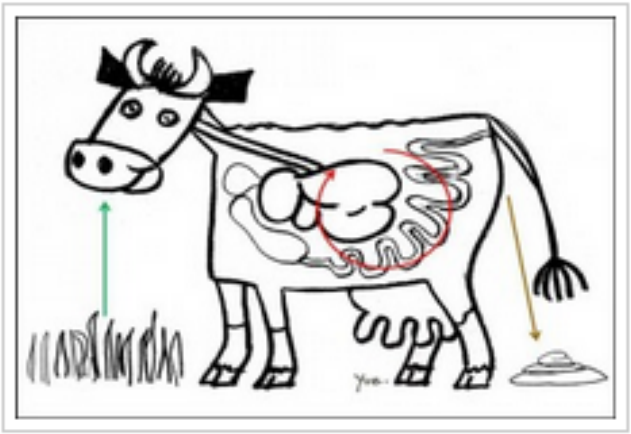
Schmutz gibt es in allen Dimensionen

IKS
Individuelle Softwarelösungen

1. Prozess: *Wie läuft das Ganze ab?*
   später mehr…
2. Wissen: *Was gibt es dabei zu beachten?*
   Clean Code, CCD etc. bieten sehr viel an
3. Handwerk: *Wie arbeite ich mit welchen Tools / Techniken effizient?*
   IDEs, APIs, CI, Sonar & CCD-Praktiken bereits im Einsatz
4. Motivation: *Warum mache ich das alles (nicht)?*
   später mehr…

Unsere Dirty-Umfrage

Was stört Sie im IT-Alltag am meisten?



Für unseren Artikel *Clean-Coding-Cosmos, Teil II* im OBJEKTspektrum möchten wir Sie gerne mit einbeziehen! Um den häufigsten Übeltätern in IT-Projekten besser auf die Spur zu kommen, stellen wir im folgenden Fragenbogen eine Reihe von Punkten vor, die evtl. auch von Ihnen als extrem behindernd in Bezug auf eine effiziente Software-Entwicklung empfunden wurden/werden.

Falls Sie einen wichtigen Aspekt vermissen, dann tragen Sie ihn bitte einfach im Textfeld "Weiterer Grund" ein. Er steht dann den folgenden Teilnehmer als weitere Antwortmöglichkeit zur Verfügung. Es sind max. fünf Kreuzchen erlaubt. Nach dem Absenden sehen Sie die prozentuale Verteilung der → bisherigen Antworten.

Vielen Dank, Reik Oberrath (IKS-GmbH) und Jörg Vollmer

http://clean-coding-cosmos.de/inquiry

Unsere Dirty-Umfrage

IKS
Individuelle Softwarelösungen

- [ ] Unklare Anforderungen
- [ ] Zu häufig & schnell wechselnde Anforderungen
- [ ] Fehlendes, unklares, unterschiedliches Qualitätsbewusstsein
- [ ] Mangelndes Verständnis für Softwareentwicklung im Management
- [ ] Projektmanagement setzt falsche Prioritäten
- [ ] Management-Druck (Überstunden, schneller entwickeln!)
- [ ] Geringe oder fehlende Testabdeckung
- [ ] Unzureichende Kommunikation im Team
- [ ] Termindruck, viele Workarounds, "Quick & Dirty"-Mentalität
- [ ] Unverständlicher Code
- [ ] Unnötige Meetings & Diskussionen
- [ ] Schlechte Teamstimmung
- [ ] Nicht machbare Aufwandsschätzungen
- [ ] Technologien unpassend/ineffizient eingesetzt
- [ ] Starre Organisationsstrukturen
- [ ] Mangelnde Fortbildungsmöglichkeiten
- [ ] Unzureichende Ausbildung der Mitarbeiter
- [ ] Schlechte Tools
- [ ] Mangelnde Rollenklärung / Rollenverständnis
- [ ] Motivationsmängel: z.B. wenig Feedback, Kritik, Lob
- [ ] Menschliche Schwächen von Teammitgliedern (mich eingeschlossen)

http://clean-coding-cosmos.de/inquiry

Die Top Ten

Antworten von 147 Softwareentwicklern:		Typ
1. Unklare Anforderungen	51.1 %	RE
2. Zu häufig & schnell wechselnde Anforderungen	38.0 %	RE
3. Schlechtes Projektmanagement	29.2 %	PM
4. Termindruck => Quick & Dirty	28,5 %	PM
5. Geringe oder fehlende Testabdeckung	27.0 %	SE
6. Unverständlicher Code (z.B. hist. gewachsen)	27.0 %	SE
7. Unzureichende Kommunikation im Team	26.3 %	KO
8. Ineffektive (Entwicklungs-)Prozesse	20.4 %	PM
9. Unnötige Meetings & Diskussionen	16.1 %	KO
10. Nicht machbare Aufwandsschätzungen	14.6 %	PM

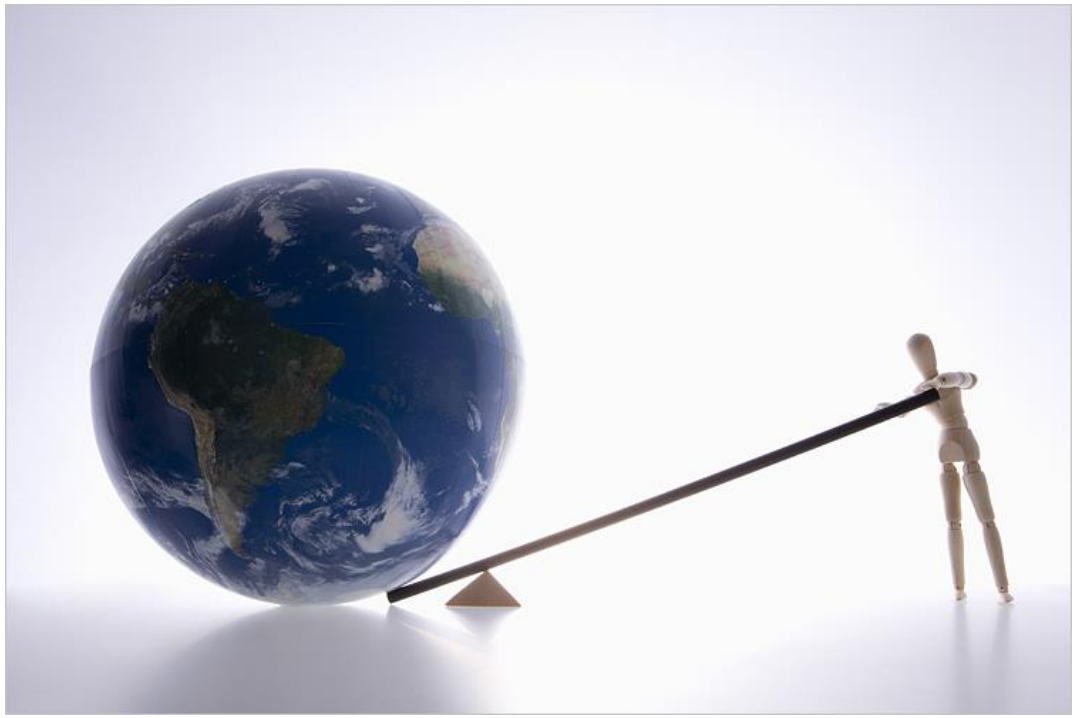
http://www.sigs-datacom.de/fileadmin/user_upload/zeitschriften/os/2014/01/oberrath_vollmer_OS_01_14_jfuz.pdf

Was behindert Entwickler am meisten?

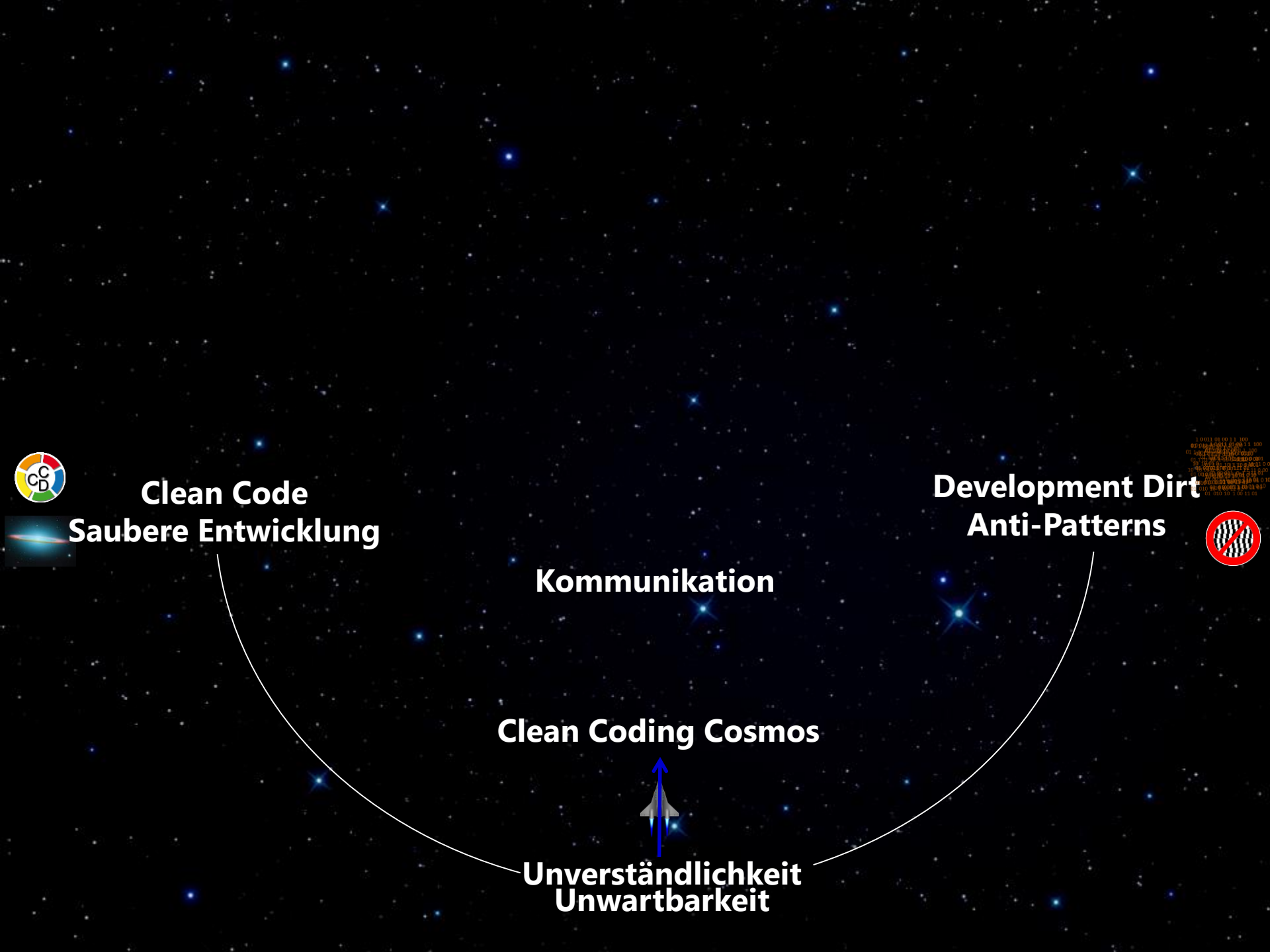
Problemtyp	%
PM	26,8
RE	19,1
Rahmenbedingungen	22,7
Technische SEP-Schwächen	18,2
Kommunikation	9,1
Teamstimmung	4,1

http://www.sigs-datacom.de/fileadmin/user_upload/zeitschriften/os/2014/01/oberrath_vollmer_OS_01_14_jfuz.pdf

Was behindert Entwickler am meisten?



Unzureichende Kommunikation!

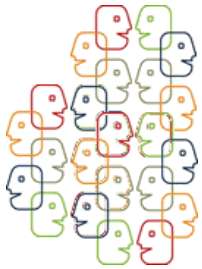

Clean Code
Saubere Entwicklung
Development Dirt
Anti-Patterns
Kommunikation
Clean Coding Cosmos
Unverständlichkeit
Unwartbarkeit

Prozess-Ebenen

- Der Entwicklungsprozess i.e.S.
  - Entwickler
  - Architekt
  - Tester
  - Product Owner
  - …

CoreTeam

- Der Rahmenprozess des ALMs
  - Domain-Experten der Fachseite
  - Anforderungsmanager (RE)
  - Architekten
  - Administratoren des Betriebs
  - Projekt- / Produkt-Manager
  - Gesamtentwicklungsleiter



Big Team

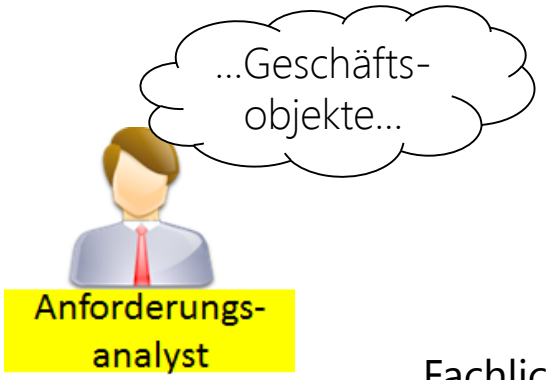
Kommunikationsebenen

Es gibt Unverständliches und Missverständliches

1. im gesprochenen Wort (Gespräche, Meetings, Vorträge, …)
2. im geschriebenen Text (Doku, Mail, Buch, Artikel, Fehlermeldungen, …)
3. im Verhalten (Sensibilität, Gestik, Mimik, Empathie, Engagement, …)
4. im Programm-Code (Kommentare, Bezeichner, Strukturen, …)

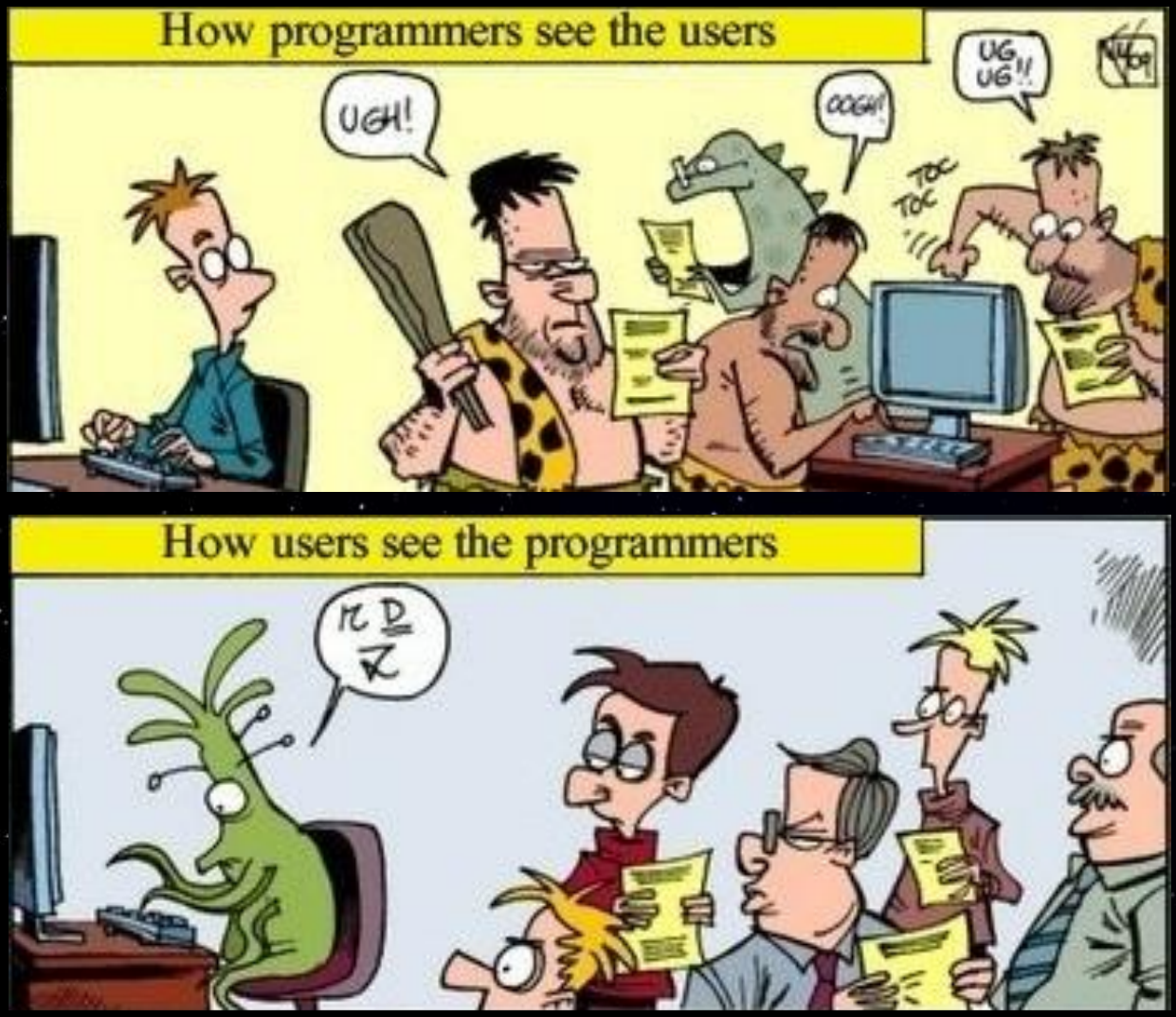
Kommunikationsbarrieren

Typische Kommunikationsprobleme bestehen zwischen Akteuren verschiedener Rollen.

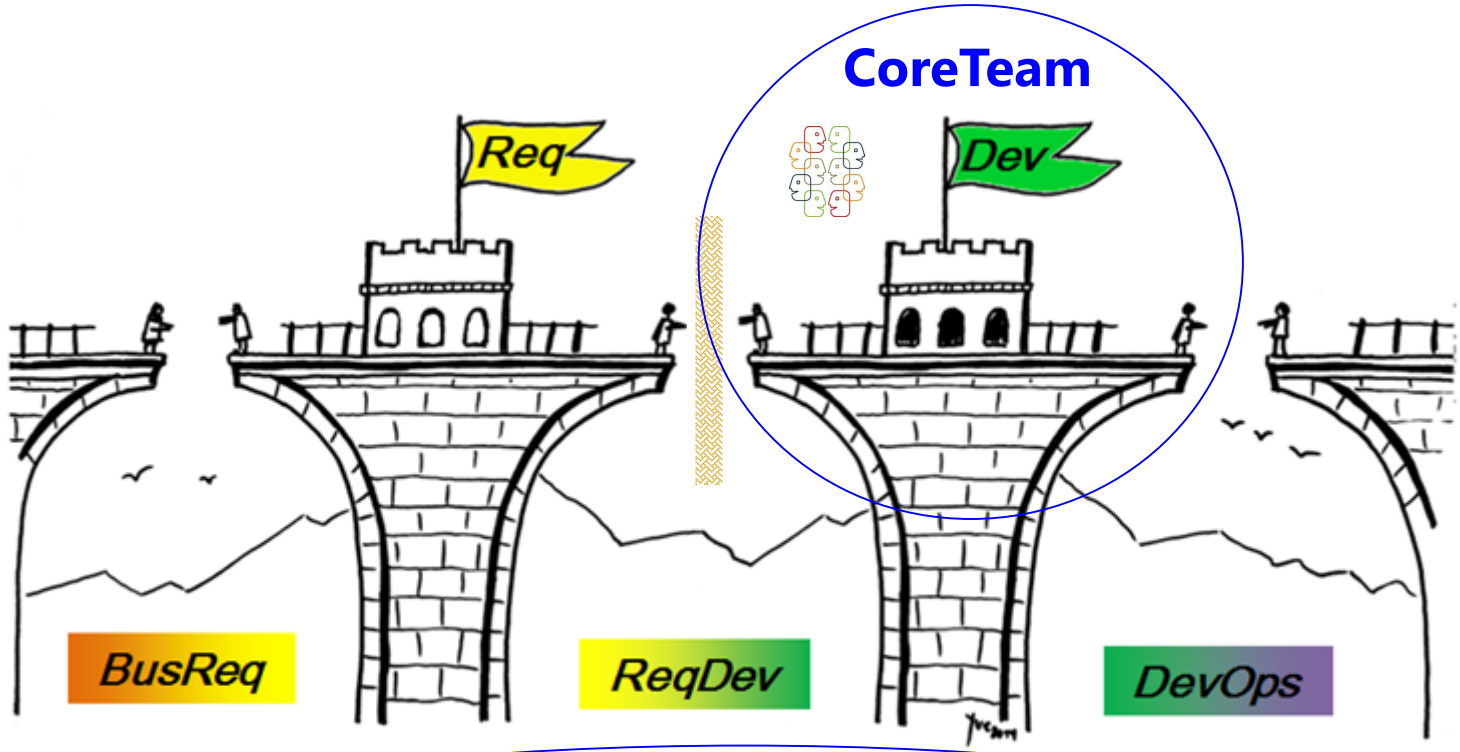


Fachliche Gedankenwelten

Technische Gedankenwelten


How programmers see the users
UGH!
OOGH!
UG UG!!
TOC TOC
How users see the programmers

Kommunikationsbarrieren




Fachexperte

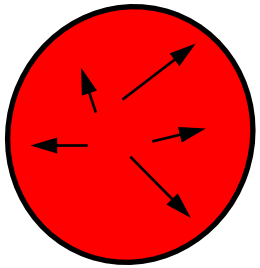

Anforderungs-analyst


Entwickler

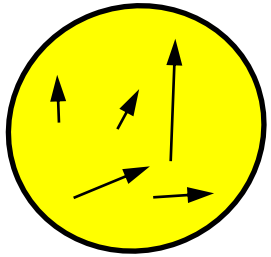

Administrator

Big Team

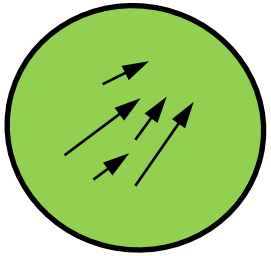
Was ist ein Team?



Haufen



Gruppe



(echtes) Team

Gefahren für Clean Communication

- Persönliche Sympathien und Antipathien führen zu Konflikten
- Gleichgesinnte verbünden sich: Es entstehen In- und Outsider
- Mangelndes Wissen und Handwerk für Kommunikation führen zu Missverständnissen
- Fehlende Kompromissfähigkeit verhindert einheitliches Vorgehen
- Überhebliches Verhalten führt zu Kommunikationsabbrüchen

Was macht ein Haufen zu einem Team?

IKS
Individuelle Softwarelösungen

Positiv

1. Anzahl der Frauen (bei 80% Optimum laut Science-Studie)
2. Soziale Sensibilität
3. Gegenseitiger Austausch & Kommunikation auf Augenhöhe

Unwesentlich

1. Durchschnitts-IQ
2. Maximaler IQ eines Mitglieds
3. Intro- oder Extrovertiertheit

Negativ

1. "Meinungsplattmacher“
2. Überempfindliche

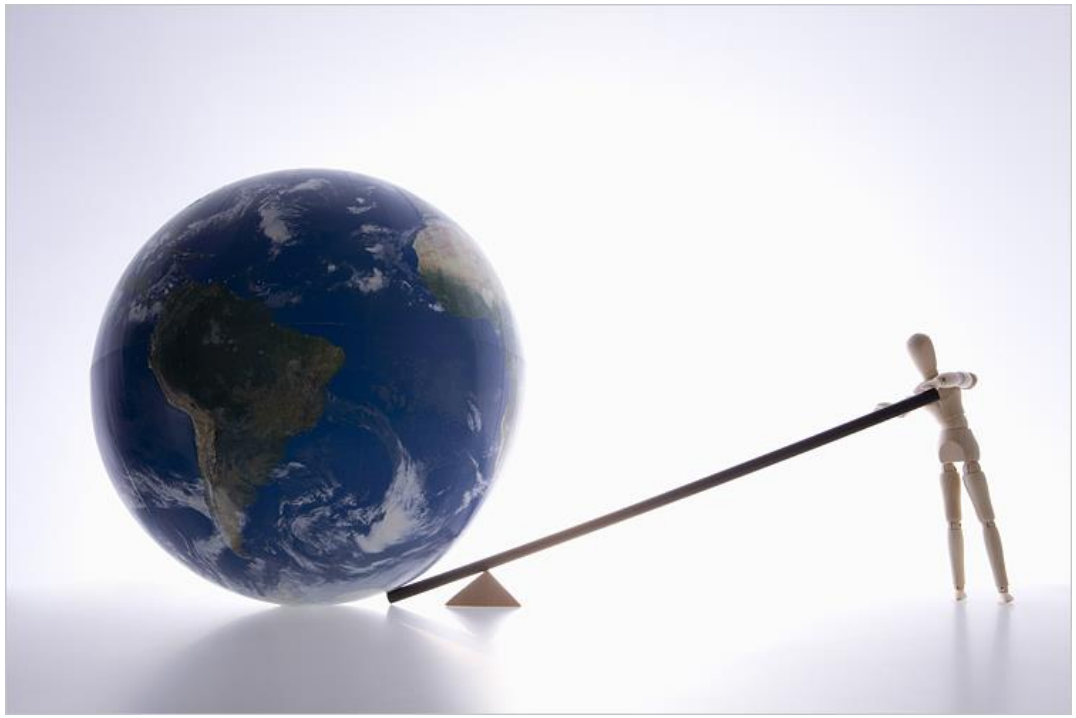
Evidence for a Collective Intelligence Factor in the Performance of Human Groups, Science 330, 686: Wooley et al. (2010)

IKS
Individuelle Softwarelösungen

Sind wir

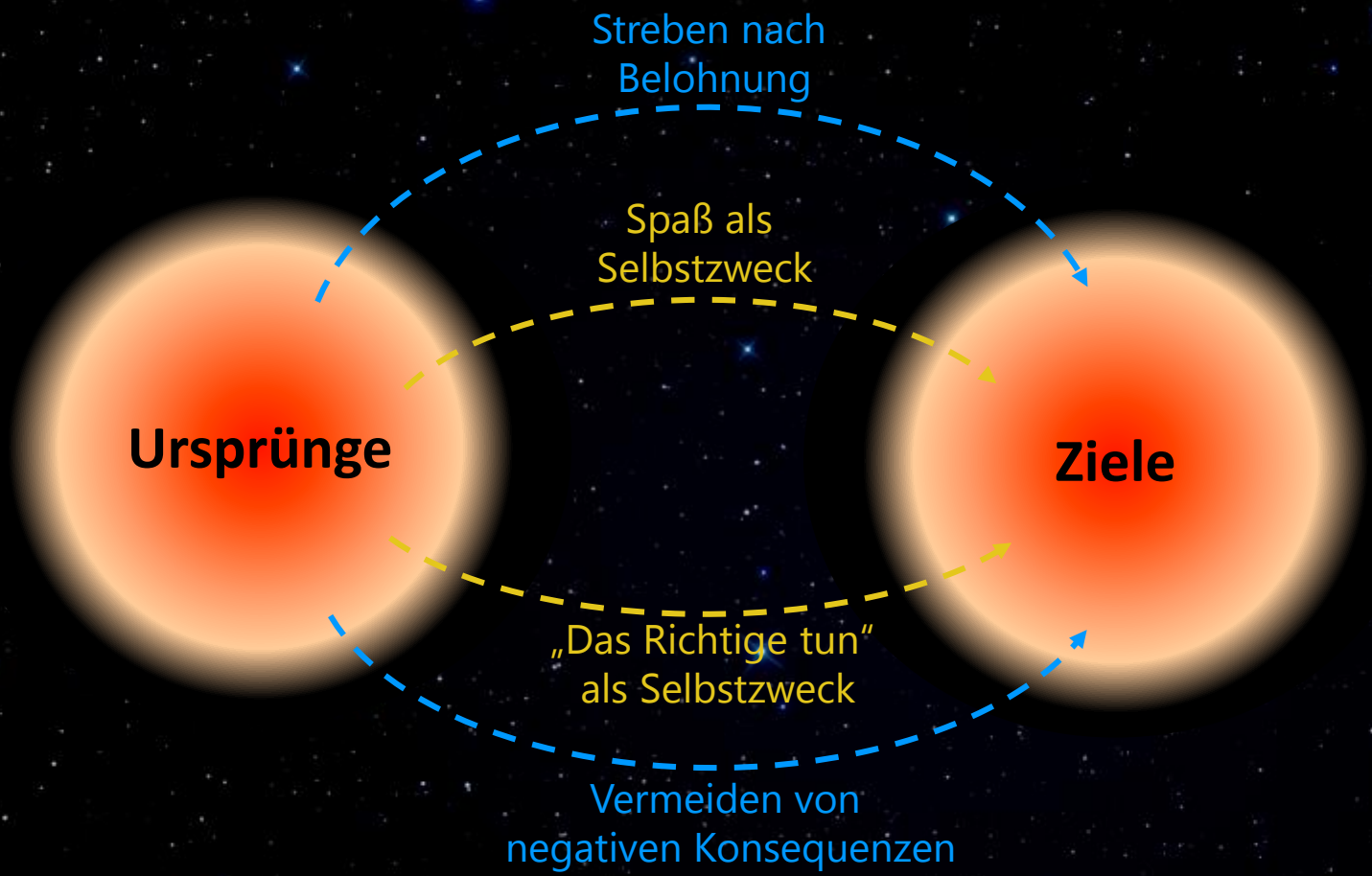
Ja klar, äh nein – also Jein!

Was lässt uns so handeln, wie wir handeln?



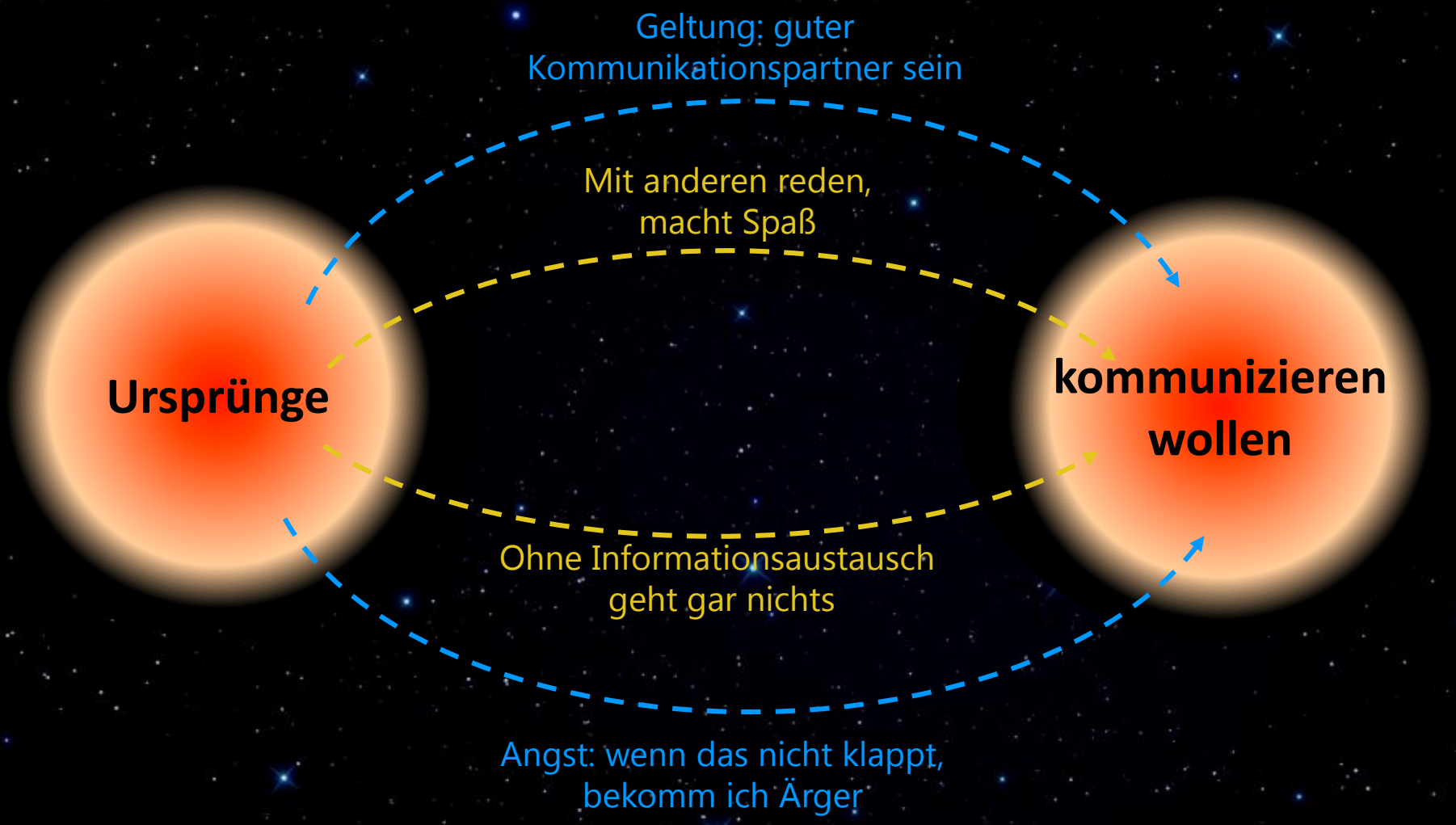
Unsere Motivation!

Motivationsformen



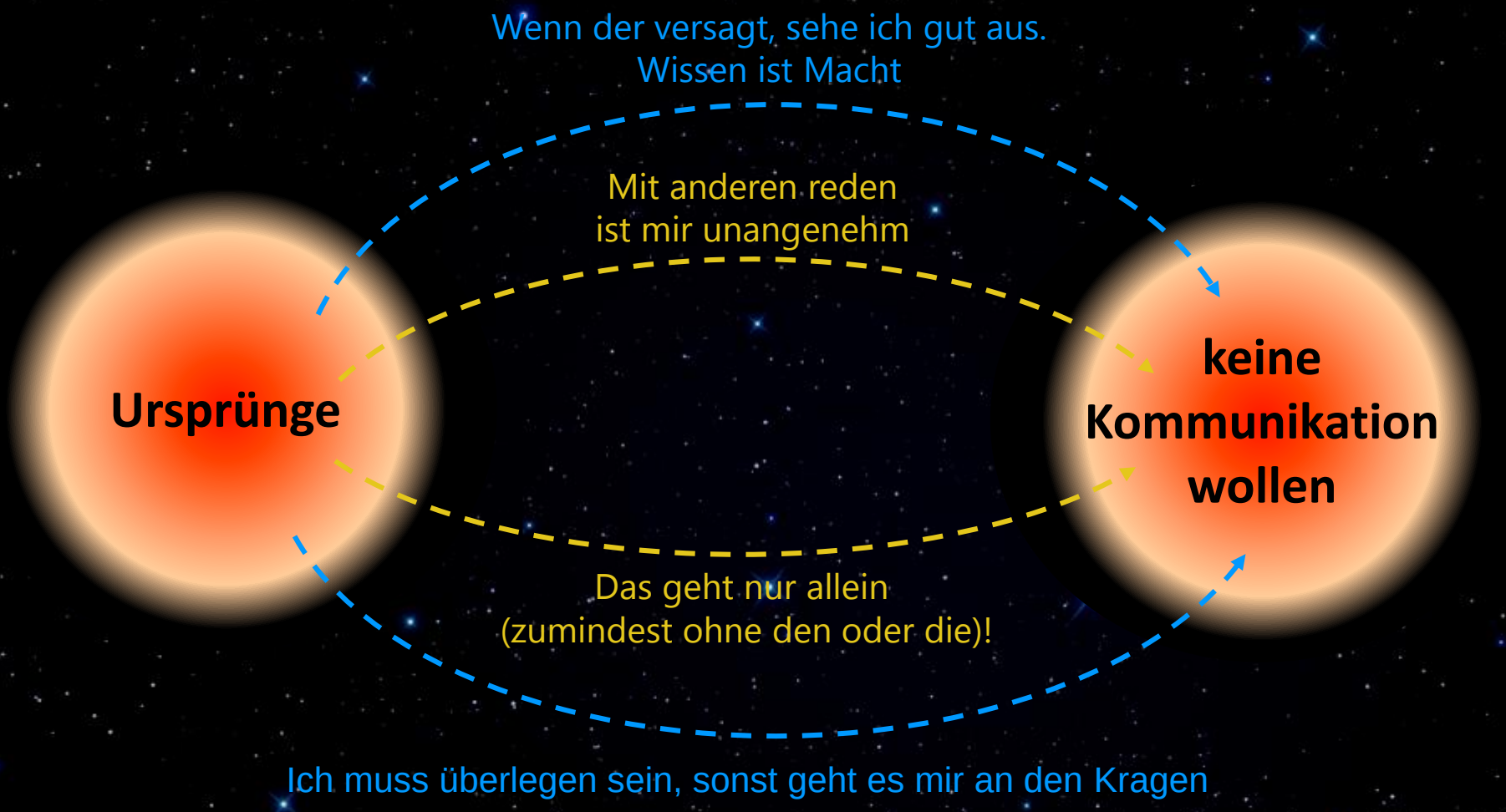
von innen kommende (Intrinsische) — **von außen kommende Motivation (Extrinsische)**

Motivation, kommunizieren zu wollen



von innen kommende (Intrinsische) – – – – – **von außen kommende Motivation (Extrinsische)**

Motivation, NICHT kommunizieren zu wollen



von innen kommende (Intrinsische) – – – – – **von außen kommende Motivation (Extrinsische)**

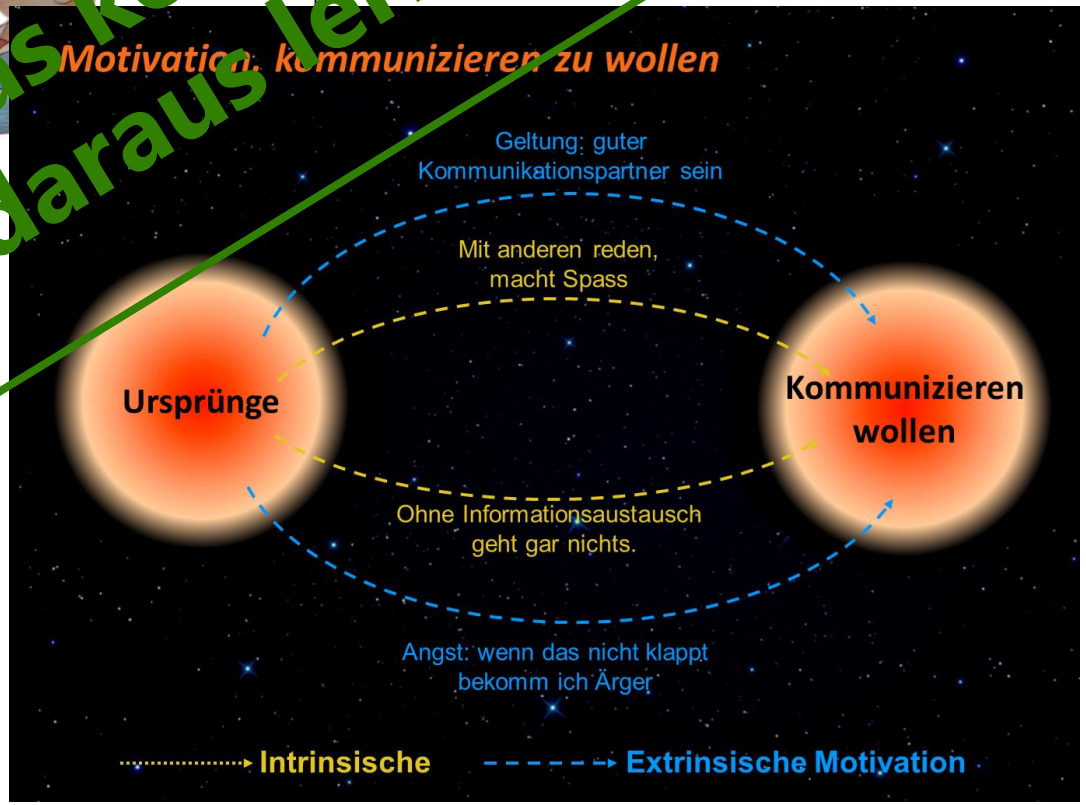
Kommunikationsebenen

IKS Individuelle Softwarelösungen

Es gibt Unverständliches und Missverständliches

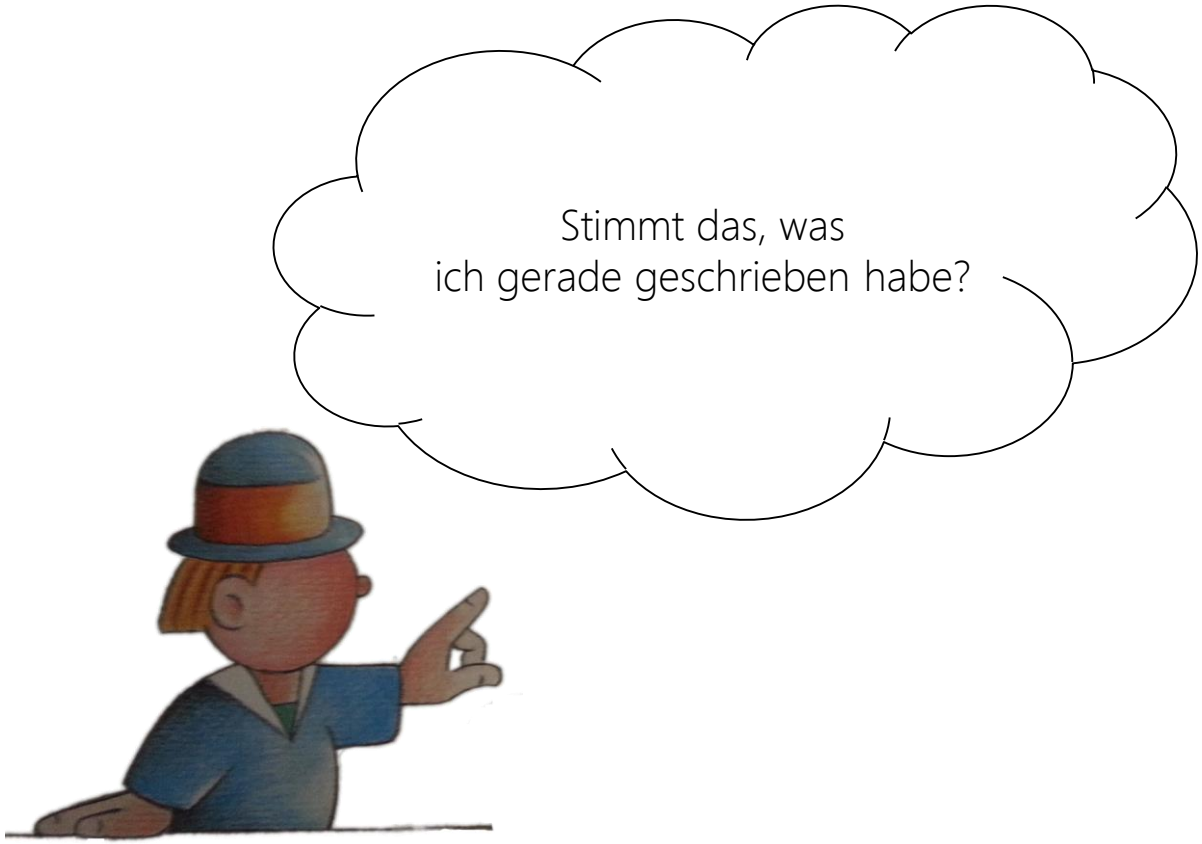
1. im gesprochenen Worte (Gespräche, Meetings, Vorträge, ...)
2. im geschriebenen Text (Doku, Mail, Buch, Artikel, Fehlermeldungen, ...)
3. im Verhalten (Sensibilität, Gestik, Mimik, Empathie, Engagement ...)
4. im Programm-Code (Kommentare, Bezeichner, Strukturen ...)

Was können wir daraus lernen?



IKS Individuelle Softwarelösungen

Motivation for Clean-Communication



Motivation for Clean-Communication

Motivation for Clean-Communication

Motivation

Clean Code
Saubere Entwicklung

Development Dirt
Anti-Patterns

Kommunikation

Clean Coding Cosmos

Unverständlichkeit
Unwartbarkeit

Motivatoren für Verhalten(-sänderungen)

Verbote

Kontrolle

Anleitung

Vorbilder

Überzeugung

Vorteil

Teamgeist

Spaß

Positive Motivation (dafür)

- Idealismus
  - in der Sache das Richtige tun
- Altruismus
  - anderen wartbaren Code zur Verfügung stellen
- Selbstwert
  - das Bewusstsein, ein Profi zu sein
- Image-Gewinn
  - Steigerung im Ansehen anderer
- Eigener praktischer Vorteil
  - mein Code ist wartbarer

Negative Motivation (dagegen)

- Der innere Schweinehund / Schlendrian
  - schnell fertig werden
- fehlende Belohnungssysteme
  - kein unmittelbarer persönlicher Nutzen
- Neue Features bevorzugen
  - Neues ist spannender
  - Funktionalität findet größere Anerkennung (Lob)
- Angst vor Änderungen
  - neue Fehler einzubauen
  - nicht richtig clean coden zu können (dann besser gar nicht)
- Collective Code Ownership
  - Verantwortungsgefühl geht verloren
- Wissen ist Macht
  - Ich teile nicht gern

Wenn Kopf und Bauch kämpfen...

Nein! Gegenbeispiel: Open-Source

Viele Open-Source-Projekte enthalten hervorragenden Code!

Kommentierungsgrad exemplarisch als einfach messbarer Indikator:

quasi 100%
quasi 100%
quasi 100%
quasi 100%

Untersuchte kommerzielle Software: **< 3%, warum?**

Unterschiede Open-Source / Kommerziell

IKS Individuelle Softwarelösungen

Open-Source-Code ist sauber, wegen ...

- **Selektion**: Unverständlicher Code überlebt die Konkurrenz auf Dauer nicht
- **Eigenes Image**: Profi sein, keine Blamage vor den Augen der ganzen Welt
- **Selbstwert**: Ich find's gut, zu "den Guten" zu gehören
- **Vorbild**: Bisheriger Code erfüllt bereits hohe Standards
- **Kontrolle**: Maintainer beurteilt streng den Code (Ablehnung durchaus üblich)

Kommerzieller Code ist unsauber, wegen ...

- **Weniger Leidensdruck**: Entwickler fragen sich untereinander bei Unklarheiten
  Open-Source-Entwickler sitzen nicht im selben Büro!
- **Keine Lobby**: Projektleiter belohnen äußere Qualität (Features / Fehlerfreiheit)
  Innere Qualität sehen und verstehen sie nicht
- **Mangelndes Vorbild**: Es ist hier überall so, soll *ich* das ändern? (Broken-Window-Symptom)
- **Fehlende Kontrolle**: Code-Reviews sind selten, die Nachverbesserungen zur Folge haben!

Was fehlt im kommerziellen Umfeld?

Open Source

Was fehlt?			
	Kontrolle	Anleitung	Vorbilder
	Kontrolle		
		Teamgeist	Spaß
		Anleitung	Vorbilder
Überzeugung	Vorteil		

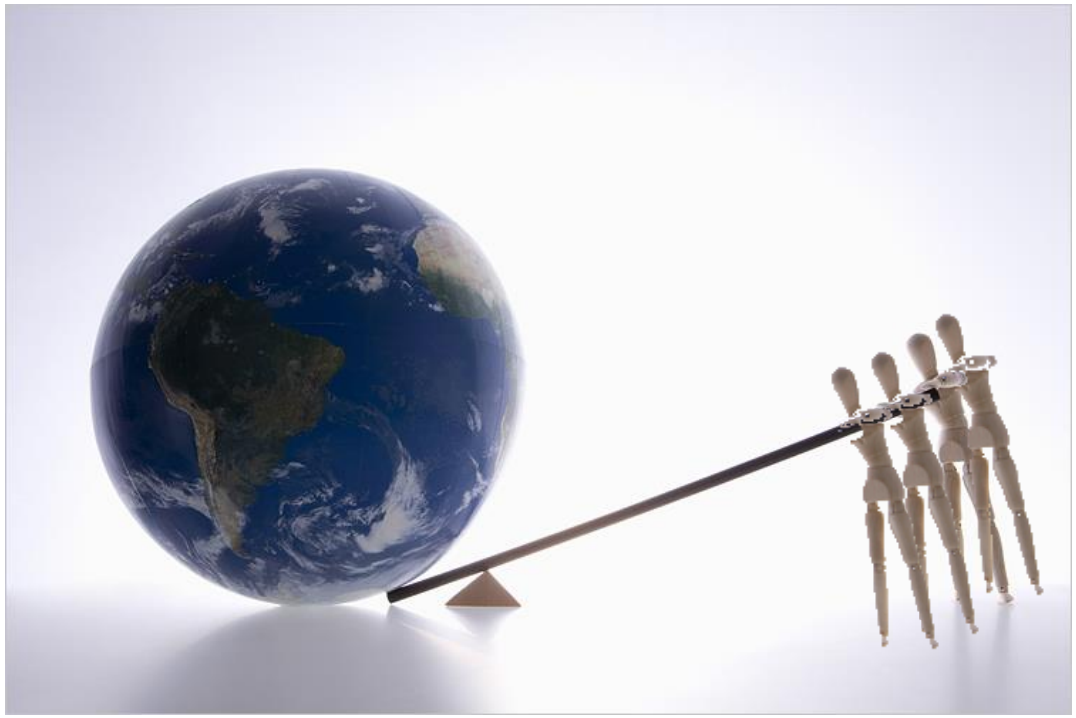
=

Der Faktor "Teamgeist"

Ein machtvolles Werkzeug

- Menschen erleben Solidarität
  - Wir haben ein gemeinsames Ziel
  - Wir sitzen in einem Boot
  - Mut zu offener, ehrlicher, direkter Kommunikation
- Menschen lernen voneinander
  - "Anders sein" des Anderen tolerieren
  - Erfahrene helfen Unerfahrenen
- Menschen geben aufeinander acht
  - Vertrauen & Sicherheit
  - Aufsicht (sozial verträgliche Form der Kontrolle)

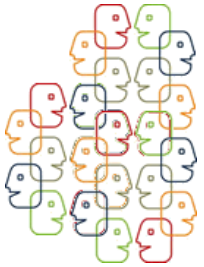
Was ist jetzt der archimedische Angelpunkt?



Der Teamgeist!

Was braucht ein Team, das effizient (clean) arbeiten möchte?

- Clean Communication (Softskill, Hardskill)
- Regelmäßige Meetings
- Verbindliche Entscheidungen treffen (können)
- Gegenseitige Unterstützung (Coaching)
- Gegenseitige Aufsicht (soziale Form von Kontrolle)
- Höherer Frauenanteil (als bisher üblich)



Big Team

Was braucht ein Entwickler-Team zusätzlich, das Clean Code möchte?

- DoD einführen: Was bedeutet bei uns "fertig"?
- DoC einführen: Was bedeutet für uns "Clean Code"?
- DoC in DoD verpflichtend integrieren (4-Augen-Prinzip)
- Pair Programming, Code Reviews, …

CoreTeam

Welche Vorgehensweisen in der IT unterstützen das?

IKS Individuelle Softwarelösungen

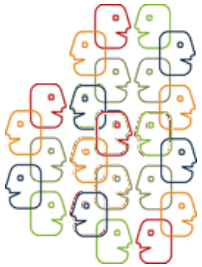
Scrum in der Entwicklung, aber

- Ist Clean verbindlich in der DoD enthalten?
- Wenn ja, wird dies konsequent kontrolliert?
- Häufig wird das frühe Umhängen der Karte belohnt!

CoreTeam

DevOps im ALM, aber

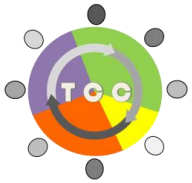
- Gehören Anforderungsmanager (RE), Domänen-Experten
- und andere Akteure auch zum Team?
- Wie kommt es zur Entscheidungsfindung?



Big Team

Unser Vorschlag für Abhilfe

Team Clean Coding



Entstand einst als Selbsthilfe in einem Core-Team.

Wir sehen darin eine Vorgehensweise,

- die für CoreTeams und BigTeams geeignet ist,
- die den DevOps-Ansatz ergänzt und unterstützt,
- die ein Plugin für Scrum darstellt,
- die aber auch klassisch "Stand Alone" funktioniert,
- in der die Ideen der Clean Code Developer vollständig eingebettet sind.

Team Clean Coding

Teamgeist

Motivation

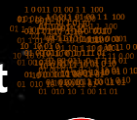
Kommunikation

Team Coding Cosmos

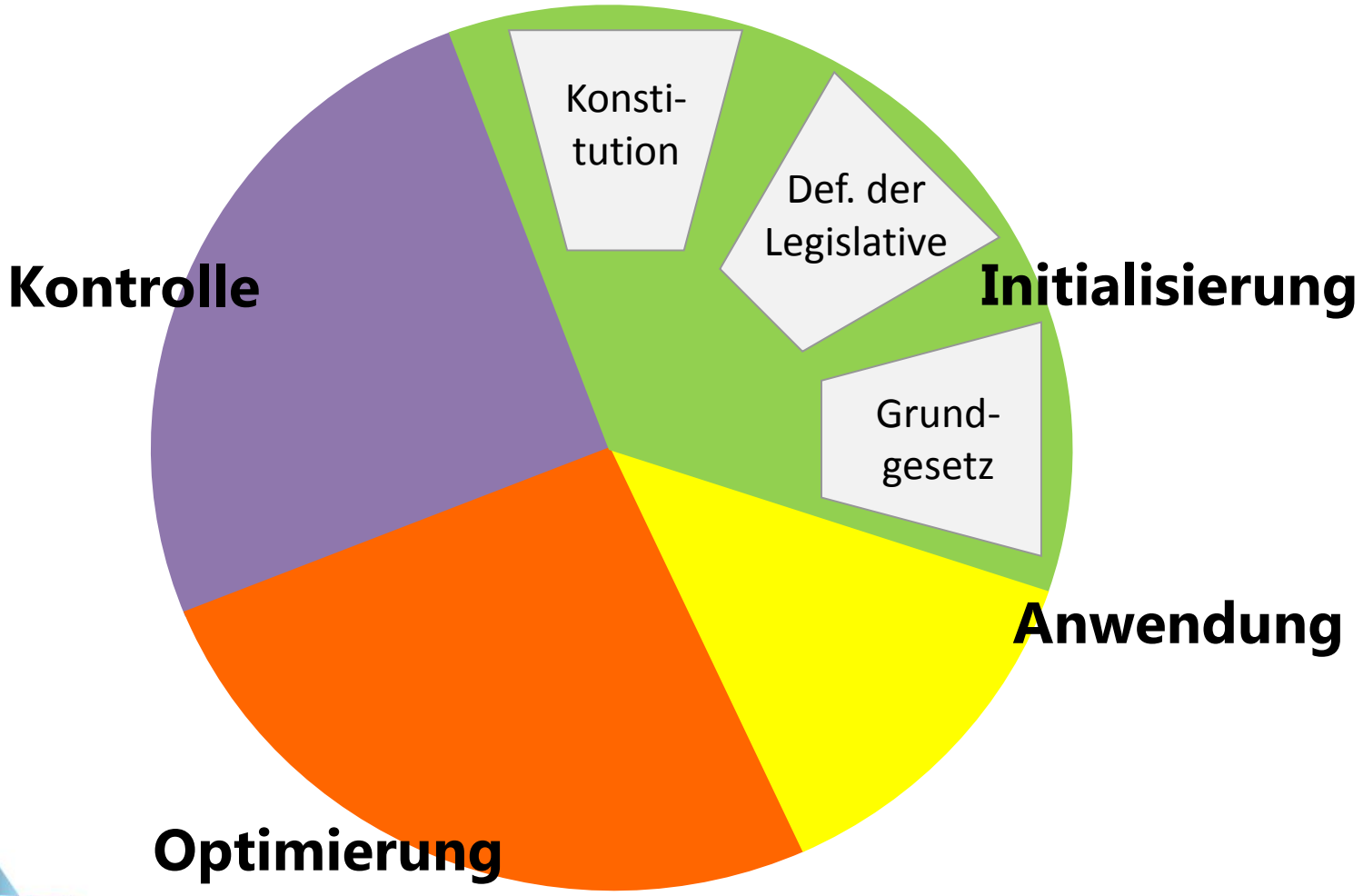
Unverständlichkeit
Unwartbarkeit

Clean Code
Saubere Entwicklung

Development Dirt
Anti-Patterns



Was ist *Team Clean Coding* genau ?



TCC-Verfassung (Konstitution)

Notwendiger Basis-Konsens

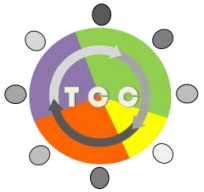
§1a ALLE Teammitglieder (inkl. Projektleitung) verpflichten sich, teaminterne Regeln und Prozesse anzuerkennen, die für ALLE gleichermaßen bindend sind.

§1b Diese Regeln und Prozesse können bei Bedarf jederzeit angepasst werden (kontinuierlicher Verbesserungsprozess).

§2 Zuerst wird ein Verfahren zur Entscheidungsfindung festgelegt (Definition der "Legislative").

§3 Die "Legislative" legt einen Satz von Basisregeln fest (das "Grundgesetz").

Exemplarisches Regel-Set des TCC



IKS
Individuelle Softwarelösungen

- Wie kommen wir zu einer Entscheidung (Legislative) ?
  - (z.B.) Thumb-Voting mit Konsent-Beschluss
- Was bedeutet für uns clean (Grundgesetz)?
  - Die CDD-Regeln werden angewendet
  - Es gibt ein Wiki und dieses enthält AM- und Architektur-Doku (CCC-Template)
  - …
- Was bedeutet für uns "Done" (Grundgesetz) ?
  - Der Code entspricht den CDD-Prinzipien
  - Kein "Done" ohne 4-Eye-Review!
  - …
- Regelmäßige *Coding Journal Clubs* ("Open Spaces", Parlament)
  - Code Review im Team
  - Besprechung von Fachliteratur
  - …

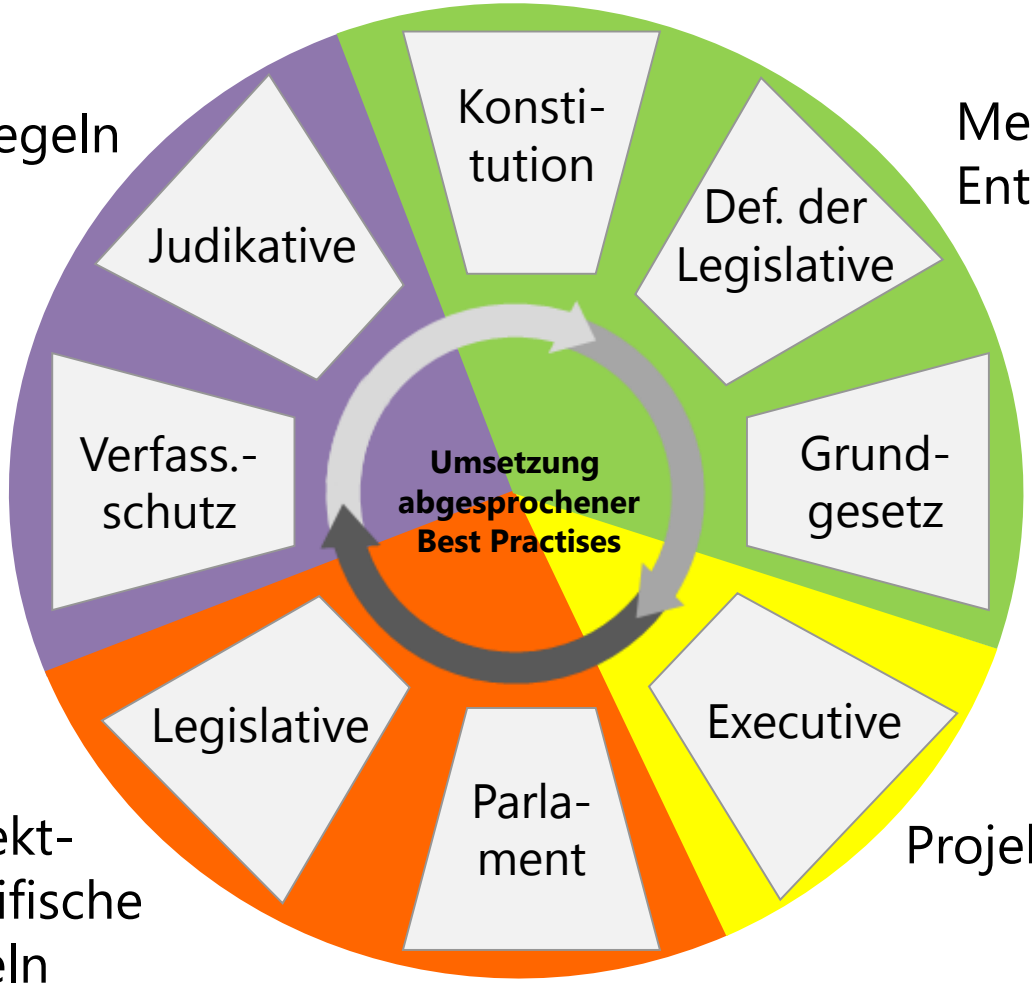
Qualitätssicherung

Basiskonsens

Initialisierung

1€-Regeln

Methode der Entscheidungsfindung



4 👁 - Review

Projekt-unspezifische Regeln

Projekt-spezifische Regeln

Projektalltag

Diskussions-forum

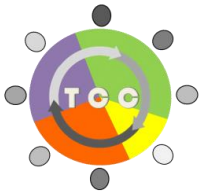
Optimierung

Anwendung

Was darf TCC nicht sein

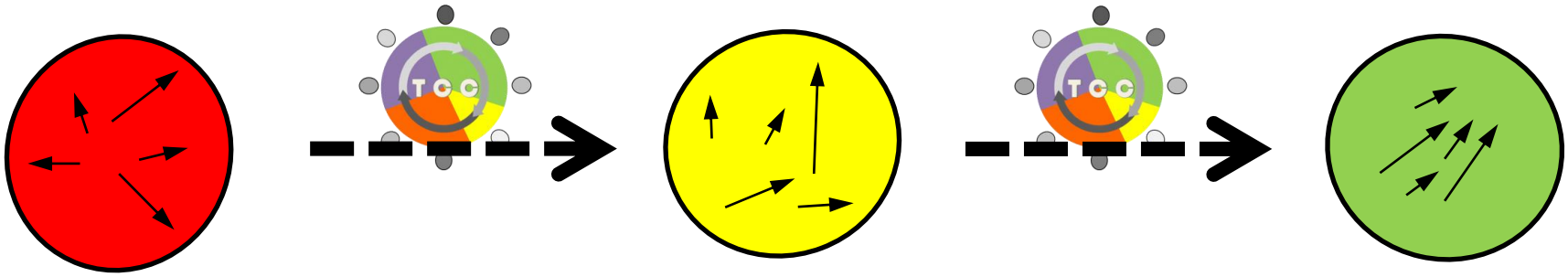
- Die Aufhebung der Gewaltenteilung
- Ein Bürgertum, das sich untereinander misstraut
- Ein Polizeistaat, der störende Bürger verhaften möchte
- Eine Staatsanwaltschaft, die Missetäter anklagt
- Ein Bürokratiemonster, das unnötigen Overhead erzeugt

Was soll TCC bewirken?



Was soll TCC bewirken?

Typische Probleme	Lösung
Selbstdarsteller & unnötige Diskussionen	Verfahren zur Entscheidungsfindung
Unterschiedliche Vorstellungen von Clean	DoC
Nichtbeachten von Clean	DoC in DoD, 4 -  - Reviews
Unterschiedliche(s) Knowhow & Erfahrung	Gegenseitiges Coaching: Pair-Programming, Journal Coding Club



Haufen **Gruppe** **(echtes) Team**

TCC plugged into Scrum

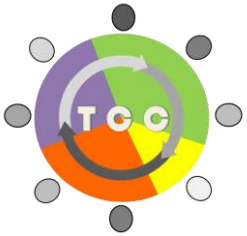
	Backlog	TODO	In Progress	In Test	Done
US 1					
US 2					

Kontrolle

	Backlog	TODO	In Progress	4-👁-Rev.	In Test	Done
US 1						
US 2						

TCC erhöht Motivation für CCD

Prinzipien & Praktiken,
→ Wertesystem, Professionalität, Selbstkontrolle



Teamwork,
→ Kommunikation, Team-Kontrolle

- Appell an Professionalität ist nicht optimal
  eine sozial verträgliche Form von Kontrolle auf Augenhöhe wirkt viel besser
- *Team Clean Coding* ist eine Sammlung von Best Practises für Teamarbeit
  und ein Zyklus kontinuierlicher Verbesserung

Wartung

Und wie bleibt nun alles schön?

Software in der Wartung: Rückfallgefahr!

Raucher hören auf, weil der Kopf ihnen sagt:

- Rauchen ist höchst ungesund
- Rauchen kostet Geld
- Alle sagen, Rauchen stinkt

Raucher werden rückfällig, weil der Bauch ihnen sagt

- Es beruhigt
- Hat etwas Geselliges
- Man ist kreativer
- Und hier guckt grad keiner

Auf Dauer clean auch ohne Team?

Entwickler arbeiten clean, weil der Kopf ihnen sagt:

- Ich bin von den Vorteilen überzeugt
- Es bewahrt uns vor "Unverständliches strengt an"
- Spart langfristig Zeit und Geld

Entwickler werden rückfällig, weil der Bauch ihnen sagt:

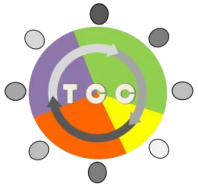
- So, läuft jetzt: Fertig! Schnell berichten und Lob ernten
- Versteht das noch jemand? Vermutlich nicht ☺
- Muss ich demnächst mal refactoren ☺

Wenn Bauch gegen Kopf kämpft…

Von Kopf bis Bauch clean

Kopf

Macht der Gewohnheit



Anleiten

Bauch

Wenn die Schlange Kaa anklopft

IKS
Individuelle Softwarelösungen

Problem:

- Kein Team(-geist)
- Keine Kontrolle durch 4-👁-Reviews
- Kein Lob für Clean

Was die Hirnvorschung rät:

- Säuberungsaktion attraktiv machen
- Life Hack: auf 50 Min. beschränken, max. dreimal täglich
- Störquellen abschalten
- Anschließend belohnen!
- Momentan wirklich keinen Nerv → TODO/FIXME annotieren!

Team Clean Coding

Teamgeist

Motivation

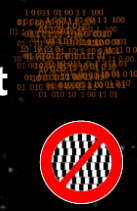
Kommunikation

Clean Coding Cosmos

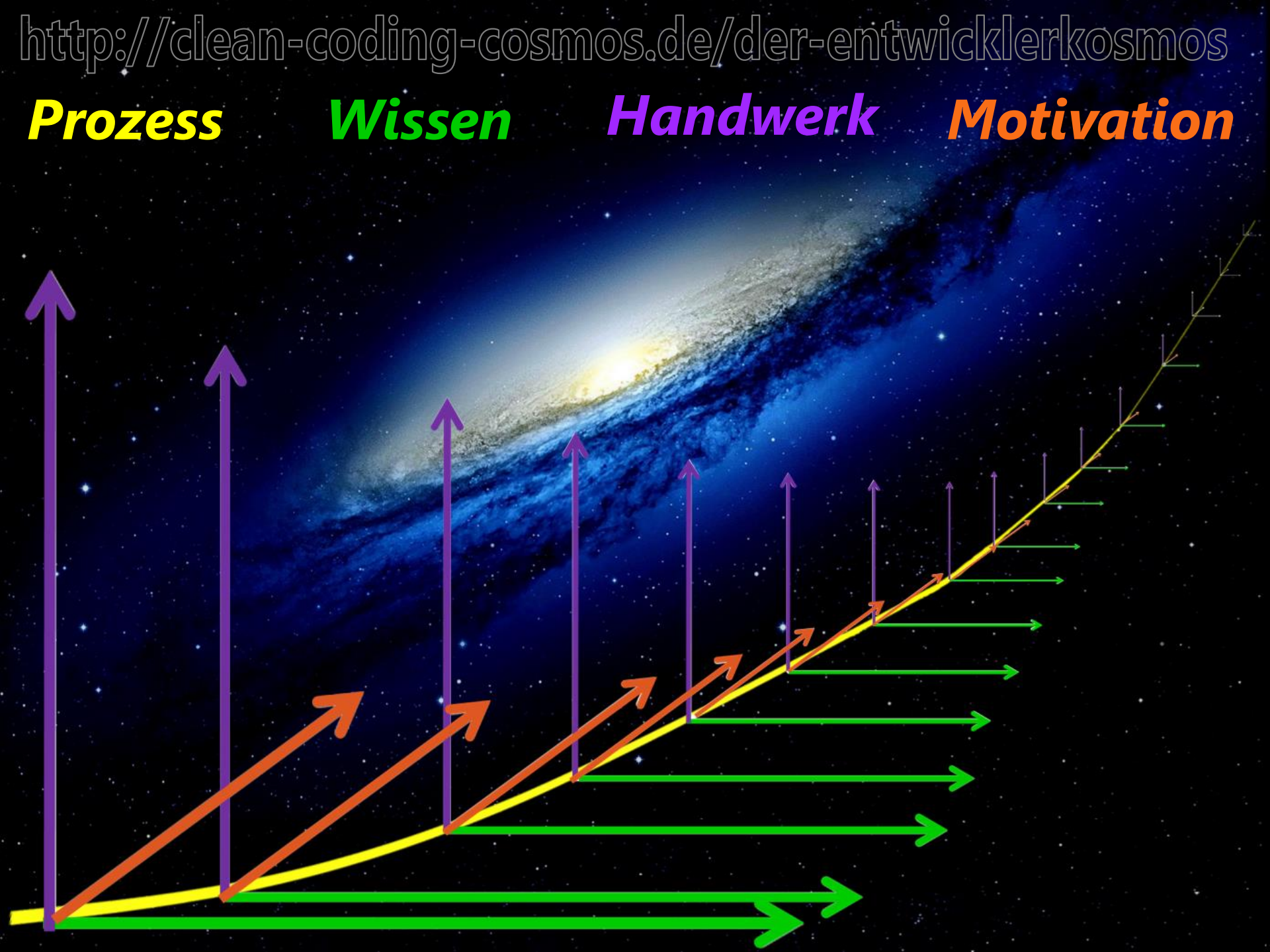
Unverständlichkeit
Unwartbarkeit

Clean Code
Saubere Entwicklung

Development Dirt
Anti-Patterns



http://clean-coding-cosmos.de/der-entwicklerkosmos



Was ist der *Clean Coding Cosmos*?

IKS
Individuelle Softwarelösungen

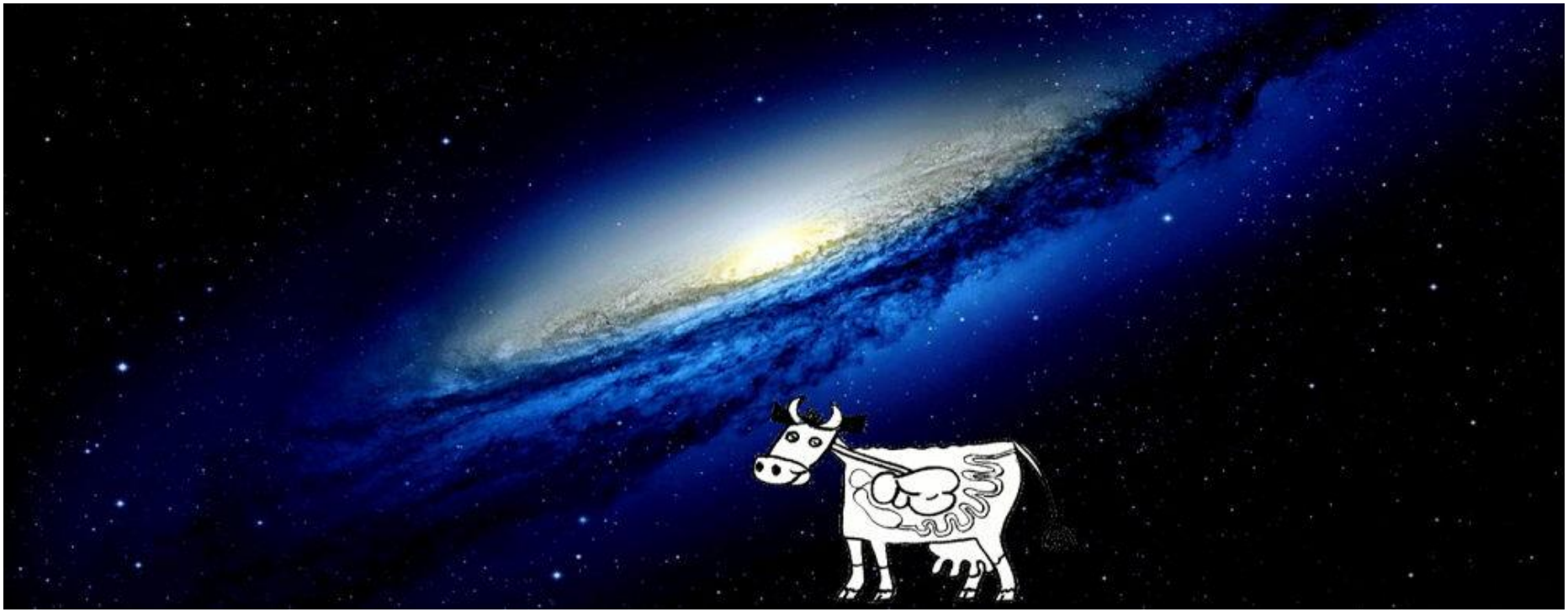
- Ein Blick auf das Big Picture der Softwareentwicklung
  (wie funktioniert das im Großen und Ganzen)

- Ein Überblick über die wichtigsten Faktoren der Softwareentwicklung,
  wie z.B. Softwarequalität und Komplexität

- Eine Knowledge Base für effektive Arbeitsweisen

IKS
Individuelle Softwarelösungen

Mehr Informationen und
Möglichkeiten zu Feedback unter
www.clean-coding-cosmos.de



Unsere Umfrage 2.0:
www.clean-coding-cosmos.de/inquiry

IKS

Individuelle Softwarelösungen

Projekte. Beratung. Spezialisten.

DIENSTLEISTUNGEN >

KOMPETENZEN | SOFTWAREENTWICKLUNG | THEMENTAGE | FACHVORTRÄGE

MOBILE APPLIKATIONEN | PROJEKTBEISPIELE

www.iks-gmbh.com

Anregung

„ Gedacht heißt nicht immer gesagt,
gesagt heißt nicht immer richtig gehört,
gehört heißt nicht immer richtig verstanden,
verstanden heißt nicht immer einverstanden,
einverstanden heißt nicht immer angewendet,
angewendet heißt noch lange nicht beibehalten. “

Konrad Lorenz (1903-89)
Österreichischer Verhaltensforscher
Nobelpreisträger 1973

- Die Motivation zum Teamwork ist der Angelpunkt, die Welt zu verändern.
  - Motivation besser zu werden
  - Motivation verantwortungsvoll zu handeln
  - Motivation zu kommunizieren
  - Motivation zur Selbstreflektion

- Im Teamgeist entsteht Motivation für Clean Communication und Clean Code

Was kann jeder von uns ganz persönlich noch besser machen?

Selbstreflexion

Passt schon!

Team Clean Coding

Teamgeist

Motivation

Clean Code
Saubere Entwicklung

Development Dirt
Anti-Patterns

Kommunikation

Clean Coding Cosmos

Unverständlichkeit
Unwartbarkeit

Fragen

TCC einführen – aber wie???

- Teamplay-Werkzeugkasten:
  - Nur einzelne Maßnahmen
  - "unter der Haube" bei Bedarf einführen
- Startup-Up-Kit:
  - reduzierter Maßnahmen-Katalog
  - zum zeitlich begrenzten Ausprobieren
- Der große Wurf:
  - Komplettes Maßnahmen-Paket
  - durch Change-Patterns vorbereiten

Change Patterns

- **Evangelist:** Selbst Change-Agent (Treiber) sein
- **Just Do it:** Die Idee selbst leben und verkörpern
- **Step by Step:** Änderungen in kleinen Schritten
- **Personal Touch:** Mit Leuten persönlich darüber reden
- **Weniger Furcht:** Negative Kritik zulassen, zuhören und lernen
- **Innovatoren / Um Hilfe bitten:** Mitstreiter finden
- **Home Town Story:** *Innovatoren* berichten von Erfolgen
- **Bridge Builder:** *Innovatoren* finden, die mit Skeptikern reden
- **Champion Sceptic:** Wortführer einer Ablehnungsgruppe Raum für Kritik geben
- **Corridor Politics:** *Personal Touch* mit Skeptikern vor Abstimmungen
- **Whisper in the General's Ear:** *Personal Touch* mit Entscheidern

http://www.sigs-datacom.de/uploads/tx_dmjournals/manns_rising_OS_01_16_UtaR.pdf