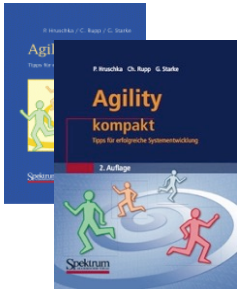
Software modernisieren, aber richtig!

Schluss mit Verschlimmbesserung

Dr. Gernot Starke



Dr. Gernot Starke

innoQ Fellow

Schwerpunkte:

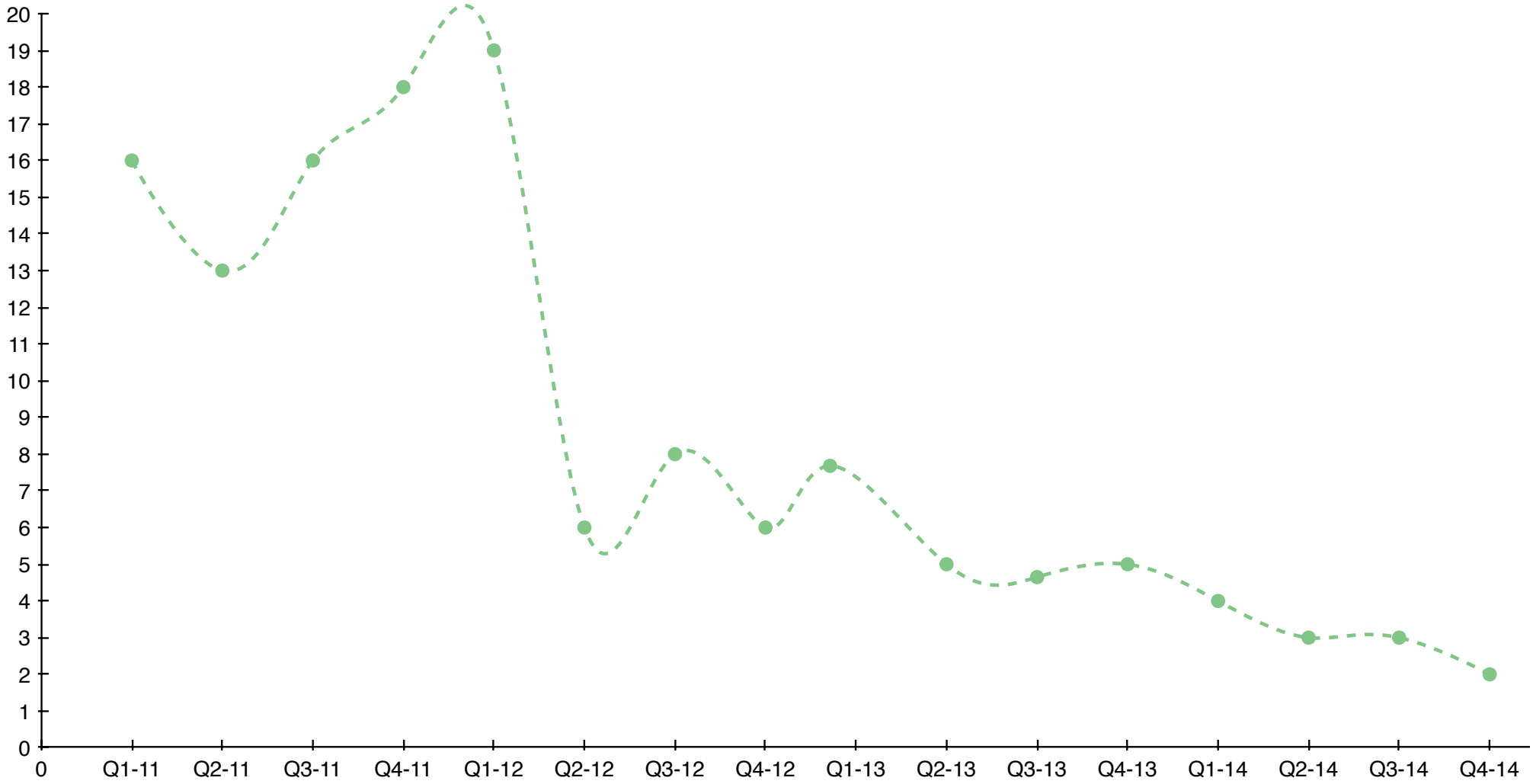
- Softwarearchitekturen
  - Entwurf, Entwicklung, Evolution
  - Modernisierung,
  - Dokumentation
- Mentoring und Coaching
- Reviews, Audits, Retrospektiven

+49 177 – 728 2570
gs@gernotstarke.de

Follow @gernotstarke

www.arc42.de

Features delivered to production



Meinung zu „System“

Management:

Zu teuer.

Zu viele Fehler.

Zu hohe Aufwände.

Zu lange Time-to-Market.

Techniker:

Zu hohe technische Schuld.

Zu wenig Verbesserung.

Technologie zu schlecht.

Zu wenig Innovation.

Zu wenig Budget.

Zu wenig Zeit.

Informatik

These:

Ausbildung fokussiert auf Neubau von Systemen

80 : 20 Regel

- 80% unserer Zeit ändern wir,
  20% bauen wir neu.

In der Ausbildung:

- 100% der Zeit lernen wir neu bauen.
- In der restlichen Zeit lernen wir ändern.

These:

Änderungen an Systemen sind durch Geld motiviert

These:

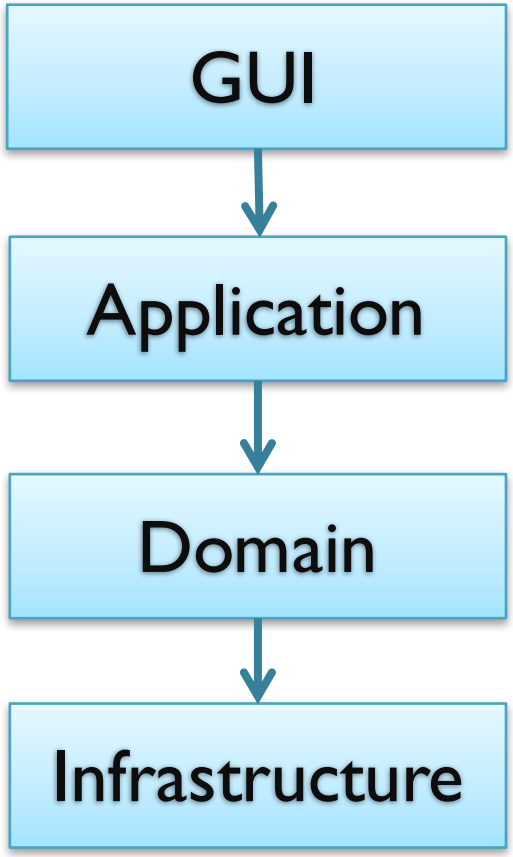
Budgetverantwortliche ignorieren Architekturprinzipien

These:

Verbesserung an Systemen

ist mehr als Refactoring einzelner Klassen

Beispiel:
Saubere Schichtung...



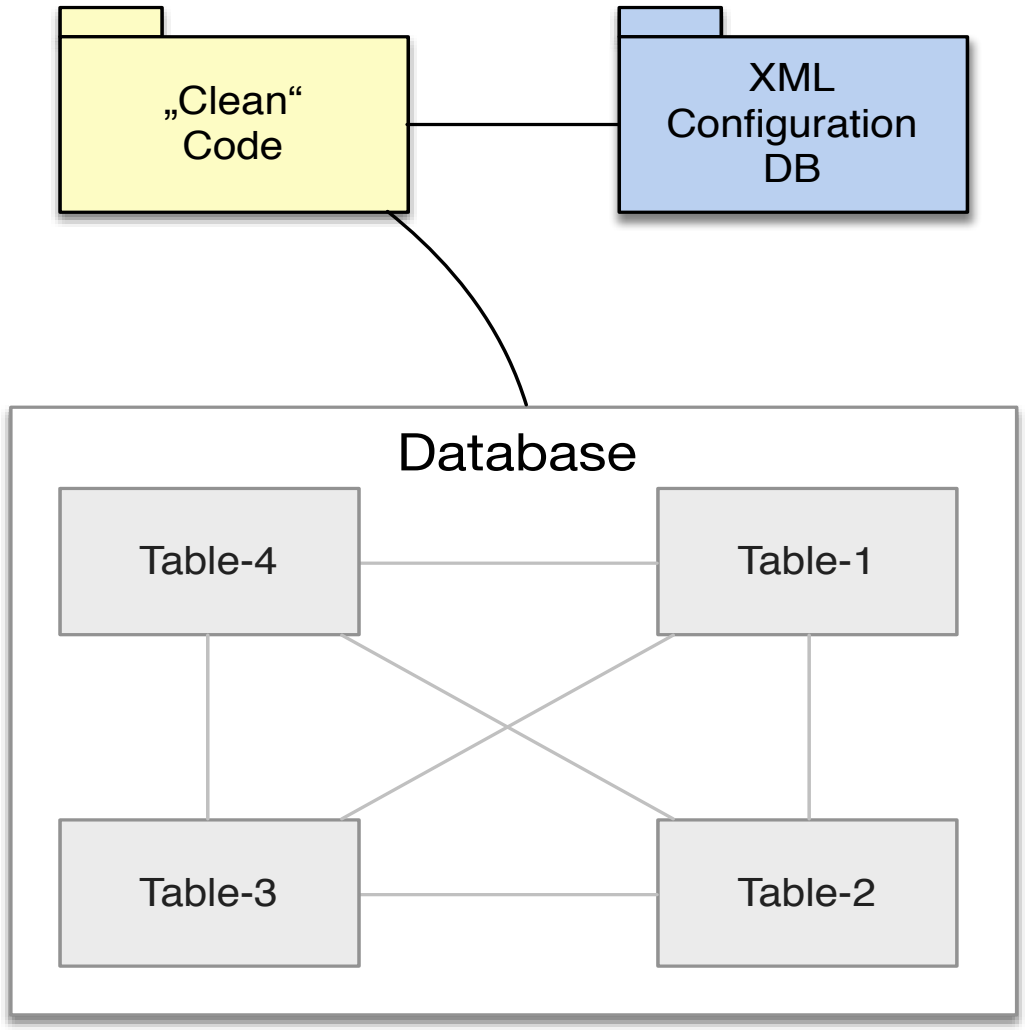
Probleme:

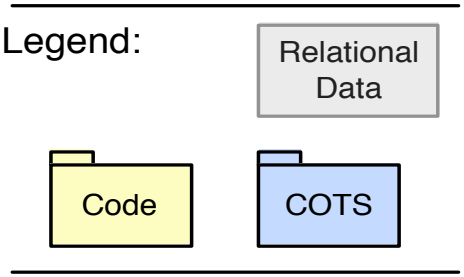
Performance

Time-to-Market

Bug-/Hotfix Zeit

Beispiel:
Saubere Schichtung, **aber**...





Peter Drucker Quotes

There is nothing so useless as doing efficiently that which should not be done at all.

Peter Drucker

Architecture Improvement Method

Gernot Starke & aim42-Contributors

http://aim42.org

Methodischer Rahmen für Optimierungs- und Veränderungsprojekte

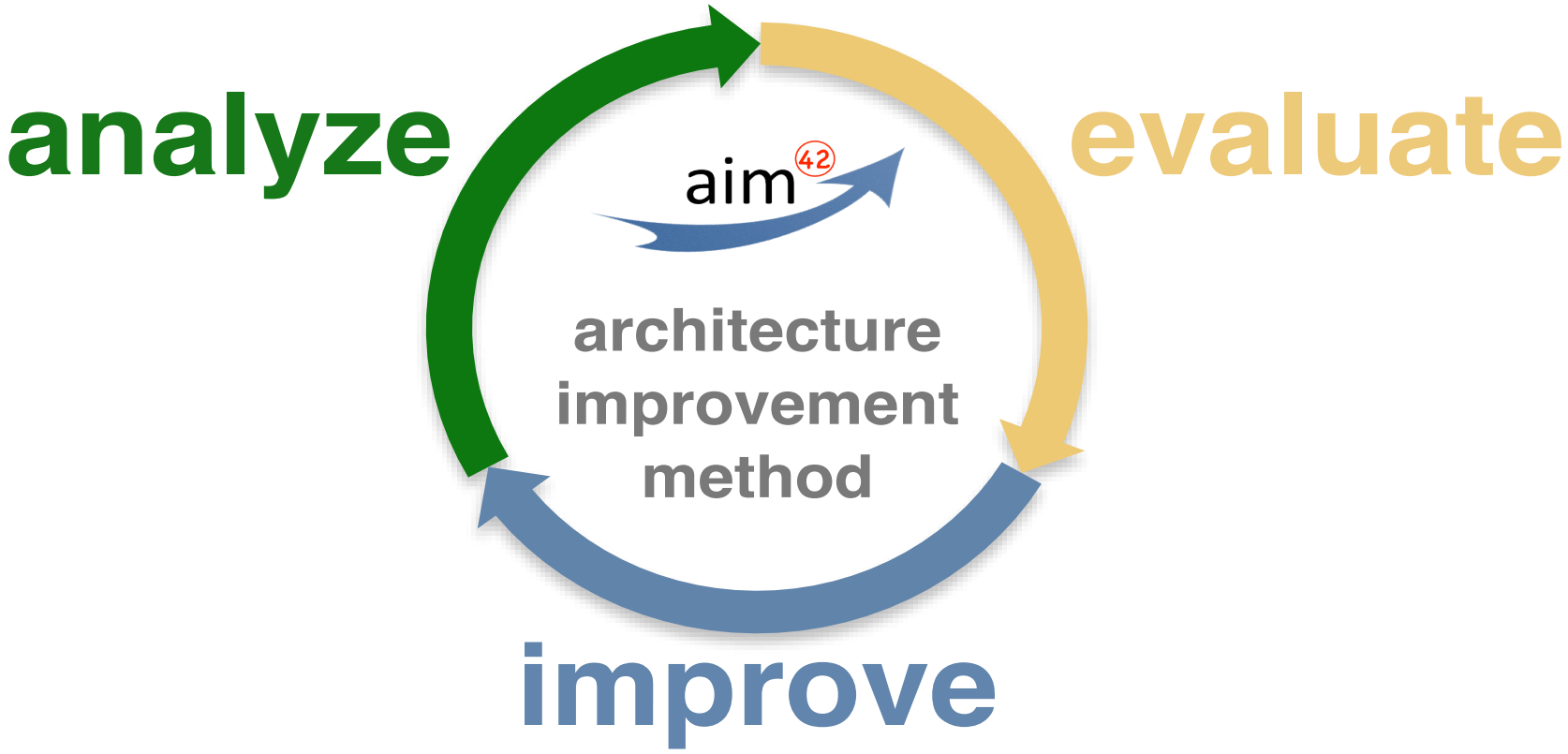
Adressiert

Business

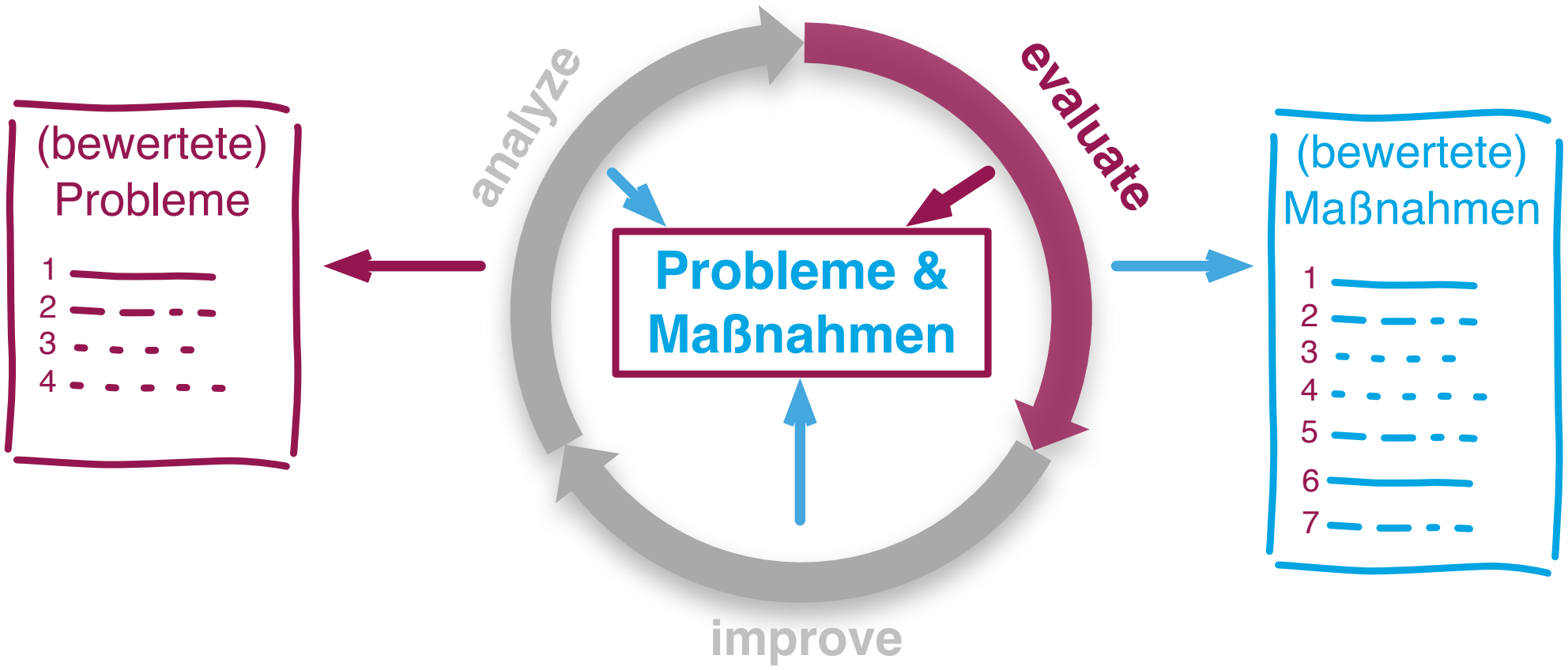
und

Technik

Flexibel & open-source

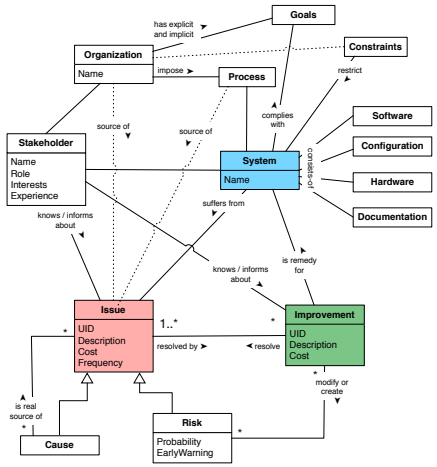

analyze
evaluate
improve
aim 42
architecture
improvement
method

Crosscutting-Activities

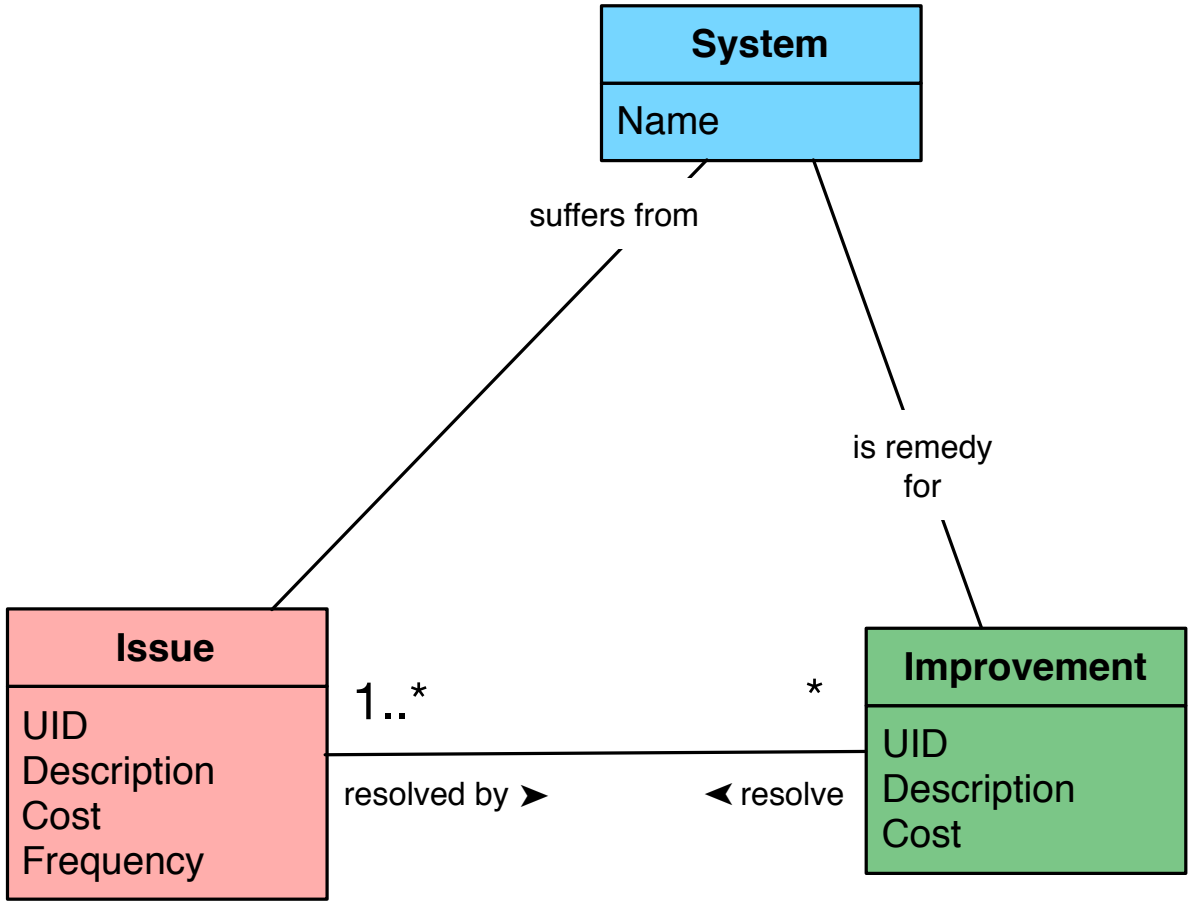


Systematisch verbessern...

Trenne „Probleme“ und „Lösungsvorschläge“



aim42 domain model



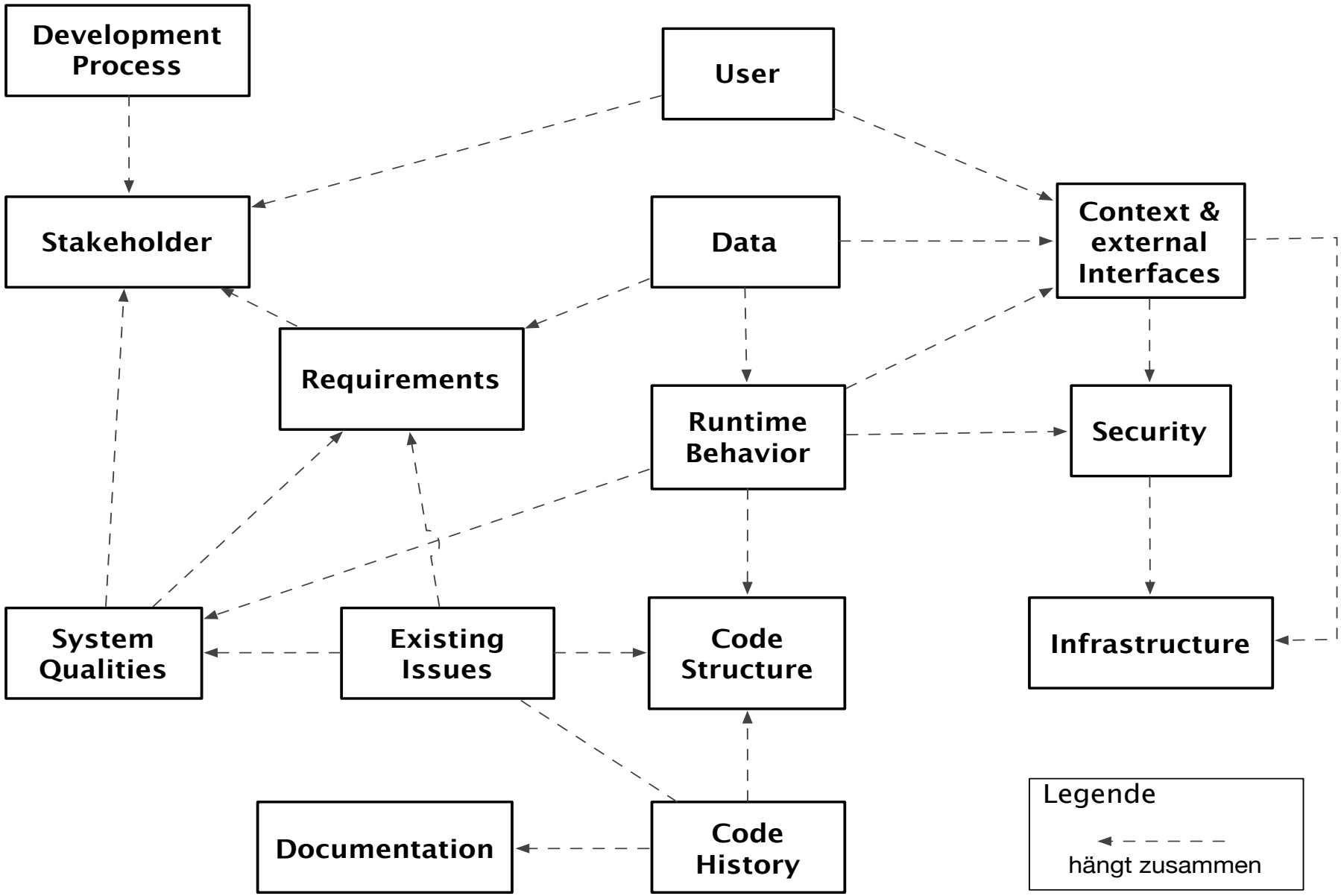
warum diese Trennung?

- richtig:
  - **wichtige** Probleme lösen

- schlechter:
  - einfache Probleme lösen

- ganz schlecht:
  - interessante Problemlösungen umsetzen
  - Aktionismus

Typische „Problemzonen“



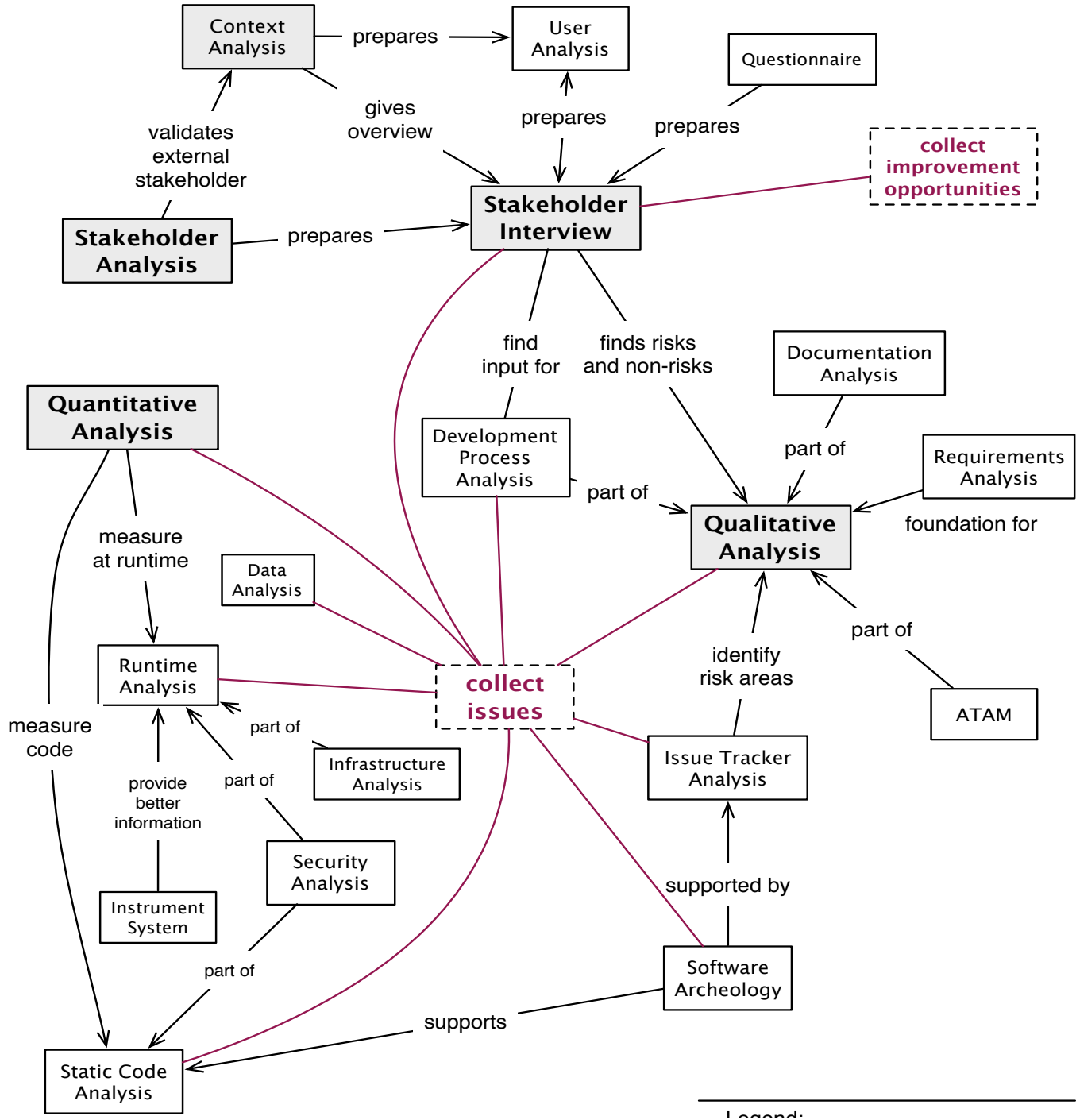
Analyse: Breitensuche...

Probleme sammeln!!

ID	Beschreibung	Häufigkeit	Wert (min)	Wert (max)	Abhilfen
PD-1	Falsche Berechnung Artikelpreis bei Kombination aus Rabattkarte, Einzelkunde und yxz-Konfiguration	zZt 3-5x pro Woche	110€	550€	V-1 + V-2
PZ-1	Zu lange Warteschlangen (queues) in Entwicklungs-prozess: waiting-time für neue Anforderungen >6W	30x / Woche	300€	15.000 €	V-3
C-1	Zeit zur Identifikation + Behebung von Laufzeitfehlern zu lang (bis zu 5 T bei kritischen Fehlern)	1x / Woche			V-4 || V-3
C-2	Zeit für kompletten Test der Anwendung > 10 T	4 x / Jahr			V-4

Werkzeuge:

- (gut) Issue-Tracker (Voraussetzung: stabile URL‘s)
- (ok) Excel, Karteikarten, Flipchart
- (schlecht) Kopf


Context Analysis
prepares
User Analysis
Questionnaire
validates external stakeholder
gives overview
prepares
prepares
collect improvement opportunities
Stakeholder Analysis
prepares
Stakeholder Interview
find input for
finds risks and non-risks
Documentation Analysis
Quantitative Analysis
Development Process Analysis
part of
part of
Requirements Analysis
Qualitative Analysis
foundation for
measure at runtime
Data Analysis
part of
identify risk areas
Runtime Analysis
collect issues
ATAM
measure code
part of
Infrastructure Analysis
Issue Tracker Analysis
provide better information
part of
Security Analysis
Instrument System
supported by
part of
Software Archeology
supports
Static Code Analysis

Nach: aim42

Statische Codeanalyse

- Kopplung
- Komplexität
- Kohäsion (inhaltlicher Zusammenhang)
- Konsistenz (identische Probleme ähnlich gelöst)
- Duplizierter Code
- Verletzung von Idiomen (Style-Checking)

Die Werkzeugfalle

Statische-Analyse-Tools finden
nur eine kleine Menge der Probleme

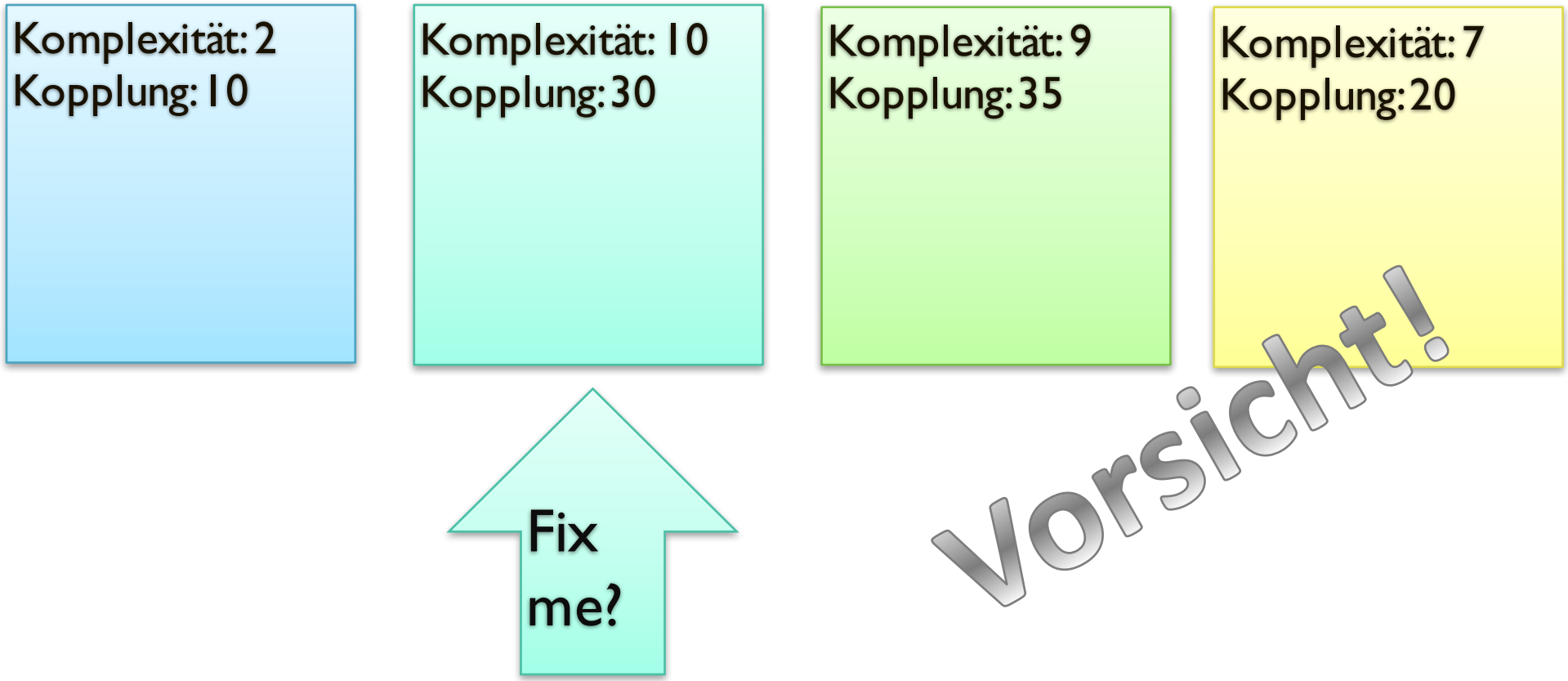
Die Mikroskop-Falle

Wenn Sie NUR im Code suchen,
werden Sie NUR DORT Probleme finden...

im Code suchen ist richtig und wichtig, aber NUR dort suchen ist fatal!

Beispiel (1)

Anmerkung: je höher desto schlechter



Beispiel (2)

Komplexität: 2
Kopplung: 10

DTFB: 0.5
Bugs: 200

Komplexität: 10
Kopplung: 30

DTFB: 2
Bugs: 30

Komplexität: 9
Kopplung: 35

DTFB: 3
Bugs: 20

Komplexität: 7
Kopplung: 20

DTFB: 8
Bugs: 15

DTFB: Days-to-Fix-Bug

Vorsicht vor isolierten Metriken!

Komplexität: 2 Kopplung: 10 DTFB: 0.5 Bugs: 200	Komplexität: 10 Kopplung: 30 DTFB: 2 Bugs: 30	Komplexität: 9 Kopplung: 35 DTFB: 3 Bugs: 20	Komplexität: 7 Kopplung: 20 DTFB: 8 Bugs: 15

DTFB: Days-to-Fix-Bug

Korrelierte Codeanalyse

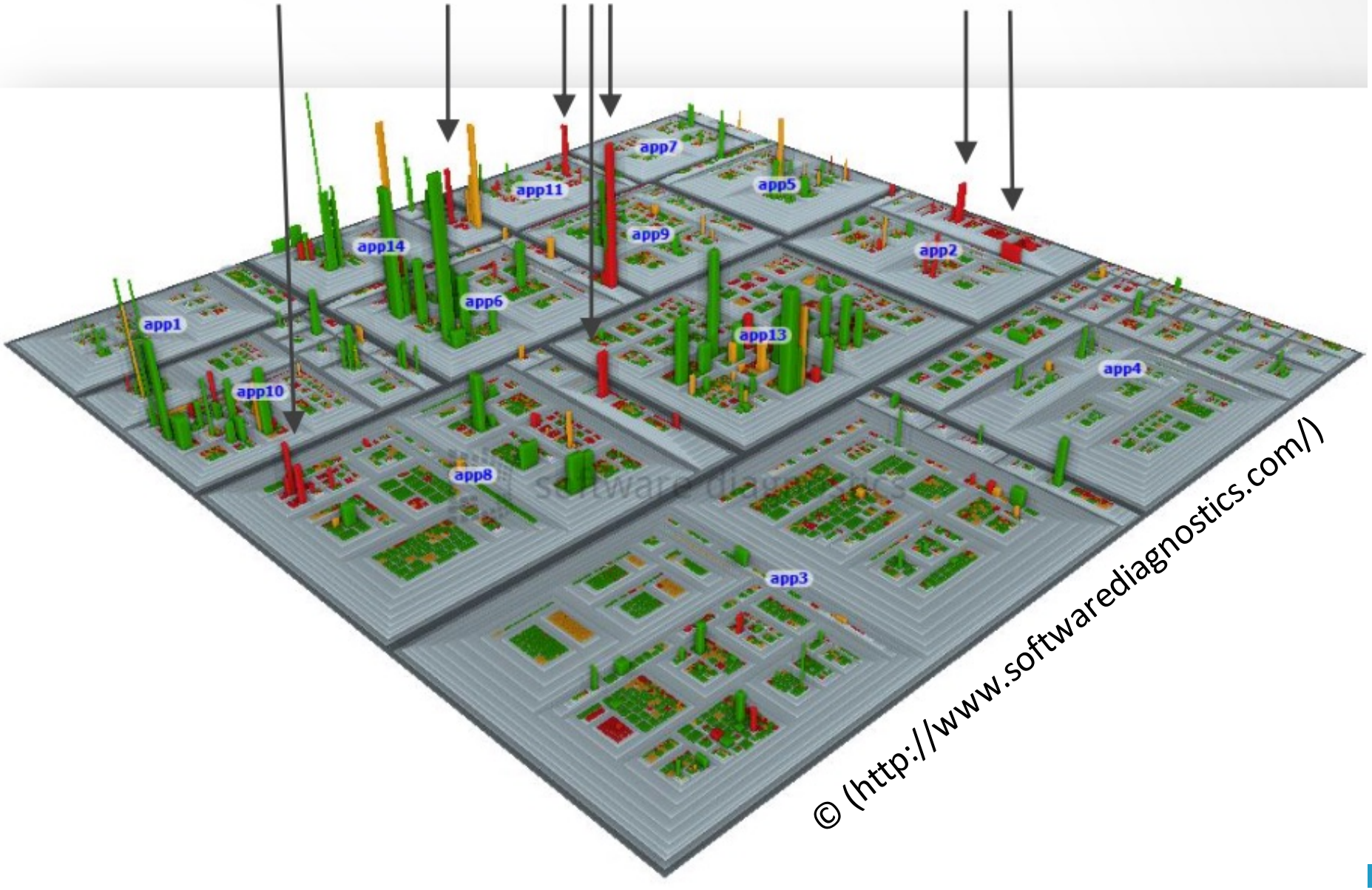
Abgleich unterschiedlicher Beobachtungen/Messungen

Beispiele:

- Fehler pro Komponente / Subsystem
- Benötigte Zeit pro Bugfix pro Komponente
- Benötigter Aufwand pro Feature pro Komponente

Korrelierte Codeanalyse (2)

Risky situation: Very complex code and only a single developer knows it. Costly effort-bombs will explode when the knowledge-having developer leaves the team.



© (http://www.softwarediagnostics.com/)

Stakeholderanalyse

Welche Personen oder Rollen?

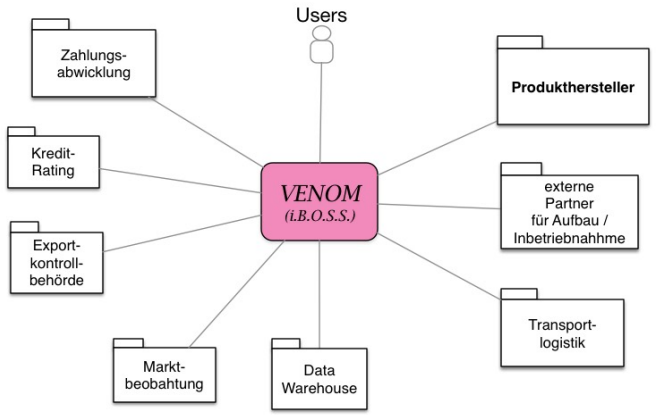
- Stakeholder kennen Probleme
  - nehmen Probleme aus ihrer Perspektive wahr (subjektiv)
  - nennen oftmals Symptome, keine Ursachen
  - äußern Vermutungen

Anwender, Entwickler, Support, Fachleute, BackOffice, Architekten, Betreiber, Auftraggeber, Tester, Admins, Projekt-/Linienverantwortliche, Controller, CEO, COO, CFO ...

- Vorsicht:
  - Gewöhnungseffekt
  - Angst vor Änderung

Kontextanalyse (1)

- Fachlicher Kontext
  - System-Scope
  - Externe Schnittstellen
    - Daten-Eingang
    - Daten-Ausgang
    - UI
    - Events
- Technischer Kontext
  - Kanäle
  - Protokolle



Prozessanalyse

- Anforderungsprozesse
  - erheben, klären, managen
- Entwicklungs- /Entwurfsprozesse
  - Architektur, Implementierung, Dokumentation
- Betrieb
  - Deployment, Rollout, Administration, Monitoring
- Management
  - Team- und Taskmanagement, Risikomanagement

warum, warum, warum...

heftiges Risiko
bzgl. Verfügbarkeit

offshore-
Dienstleister

Abnahme erteilt,
Basis „Funktionsprüfung“

Abnahmeprozess
auf Durchsatz optimiert

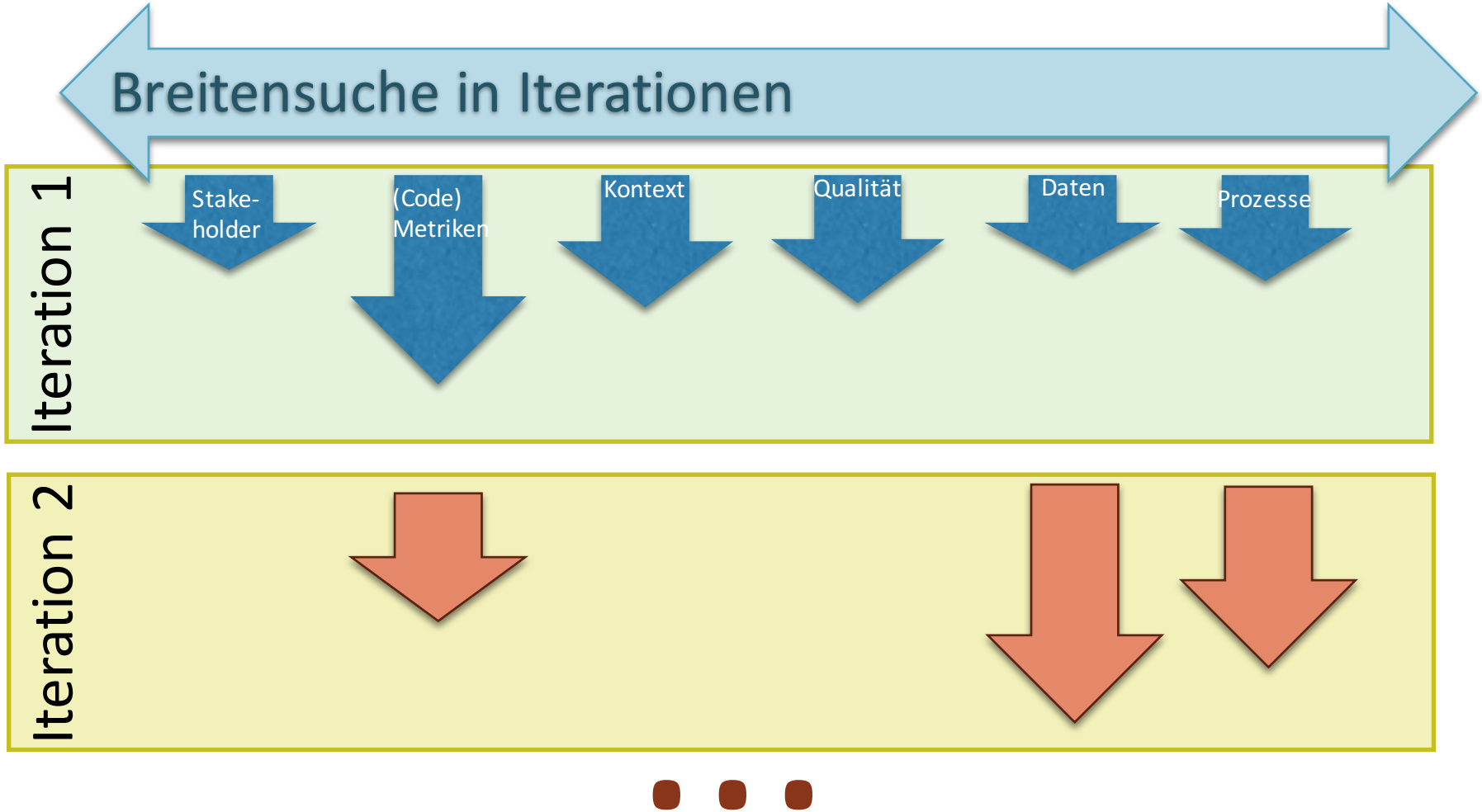
Laufzeitanalyse

- Debugger
- Logfile-Analyse

- Performance-Analyse, Profiling
- Ressourcenverbrauch zur Laufzeit

- Analyse von Benutzerverhalten
- Analyse fachlicher Abläufe

Analyse


Breitensuche in Iterationen
Iteration 1
Stake-holder
(Code) Metriken
Kontext
Qualität
Daten
Prozesse
Iteration 2

Management überzeugen

Management überzeugen...

- Problem Cost ermitteln
  - Technische Probleme haben einen Preis
  - Schätzen mit expliziten Annahmen

Beispiel: Mehraufwand „Heterogenität“

- Technologiezoo: System aus >20 Subsystemen in 8 Technologien
- Techniker: „aufwändig, komplex“
- Management: was kostet das?

Kosten der (übertriebenen) Heterogenität

- Mehraufwände in Lebenszyklus-Phasen schätzen
  - Analyse, Architektur, Implementierung, Test, Betrieb
  - Unterstützt durch reale Aufwandsmessungen

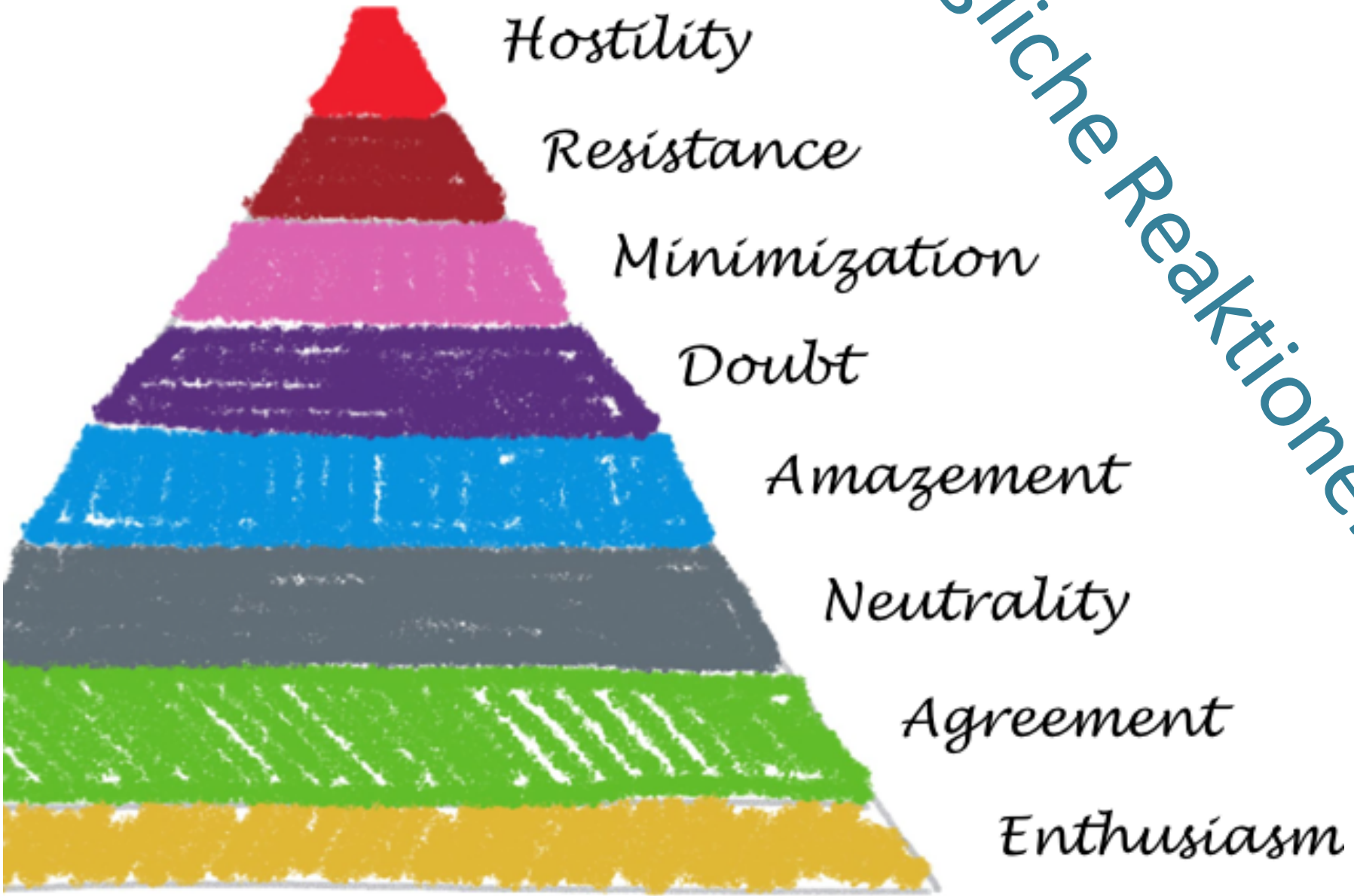
Mehrkosten „Heterogenität“ [20%..2%]

	Anteil	Mehraufwand min	Mehraufwand max	1.000 € min	max	
					1.017,78 €	1.204,56 €
Requirements	**7%**			70 €	70,00 €	70,00 €
Design/Architektur	**6%**			60 €	60,42 €	61,20 €
10% Zusatzaufwand Schnittstellen		5%	15%		0,30	0,90
10% übergreifende Entscheidungen		2%	5%		0,12	0,30
80% Sonstiges						
Programmierung	**12%**			120 €	122,40 €	145,68 €
2% Setup, Updates-Umgebung		5%	100%		0,12	2,40
2% Einarbeitung, Recherche		5%	20%		0,12	0,48
10% Fehlersuche, Testing		3%	100%		0,36	12,00
5% Effiziente Lösung von Detailproblem		-10%	-40%		-0,60	-2,40
10% Lösung von Standardproblemen		10%	50%		1,20	6,00
20% Team-interne Abstimmung		5%	30%		1,20	7,20
51% Sonstiges						
Integration / Test	**8%**			80 €	83,40 €	113,80 €
5% Komponenten integrieren		5%	100%		0,20	4,00
30% Integrationstests durchführen		5%	50%		1,20	12,00
20% Integrationstests auswerten		10%	50%		1,60	8,00
10% Testumgebung bereitstellen/erhalten		5%	80%		0,40	6,40
35% Sonstiges						
Maintenance / Operations	**67%**			670 €	681,56 €	813,88 €
3% Vorhalten von Entwicklerkapazität		5%	20%		1,01	4,02
5% Entwickler finden/einarbeiten		10%	30%		3,35	10,05
1% Versions- und Security-Updates		3%	10%		0,20	0,67
1% Auswahl / Beschaffung Laufzeitumgebungen		10%	100%		0,67	6,70
3% Konfiguration, Installation		5%	70%		1,01	14,07
0,50% Monitoring, Logging		5%	10%		0,17	0,34
5% Fehlersuche und -Behebung		1%	100%		0,34	33,50
2% Skalierung/Clustering		5%	15%		0,67	2,01
1% Paketierung, Deployment-Vorbereitung		2%	10%		0,13	0,67
30% Erweiterungen/Änderungen vornehmen		2%	30%		4,02	60,30
49% Sonstiges						

Die Relativitäts-Falle

Des Einen Problem ist des Anderen Freund

mögliche Reaktionen...



profitieren vom Problem,
haben das Problem erschaffen,
greifen Sie und Ihre Vorschläge an

sind am Problem schuld,
stehen zu Ihnen in Konkurrenz,
greifen Ihre Kompetenz an

Hostility

Resistance

Minimization

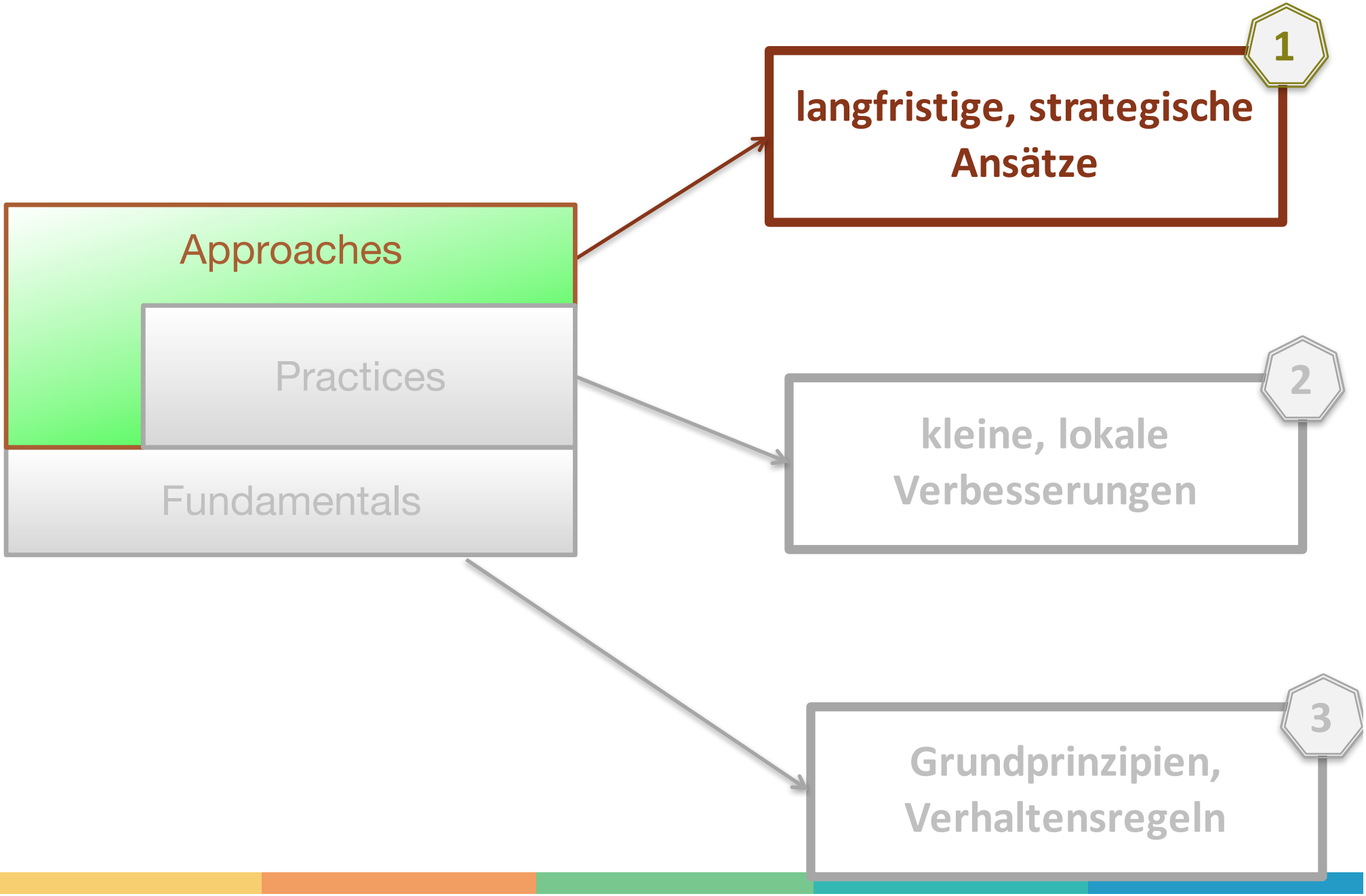
Doubt

Amazement

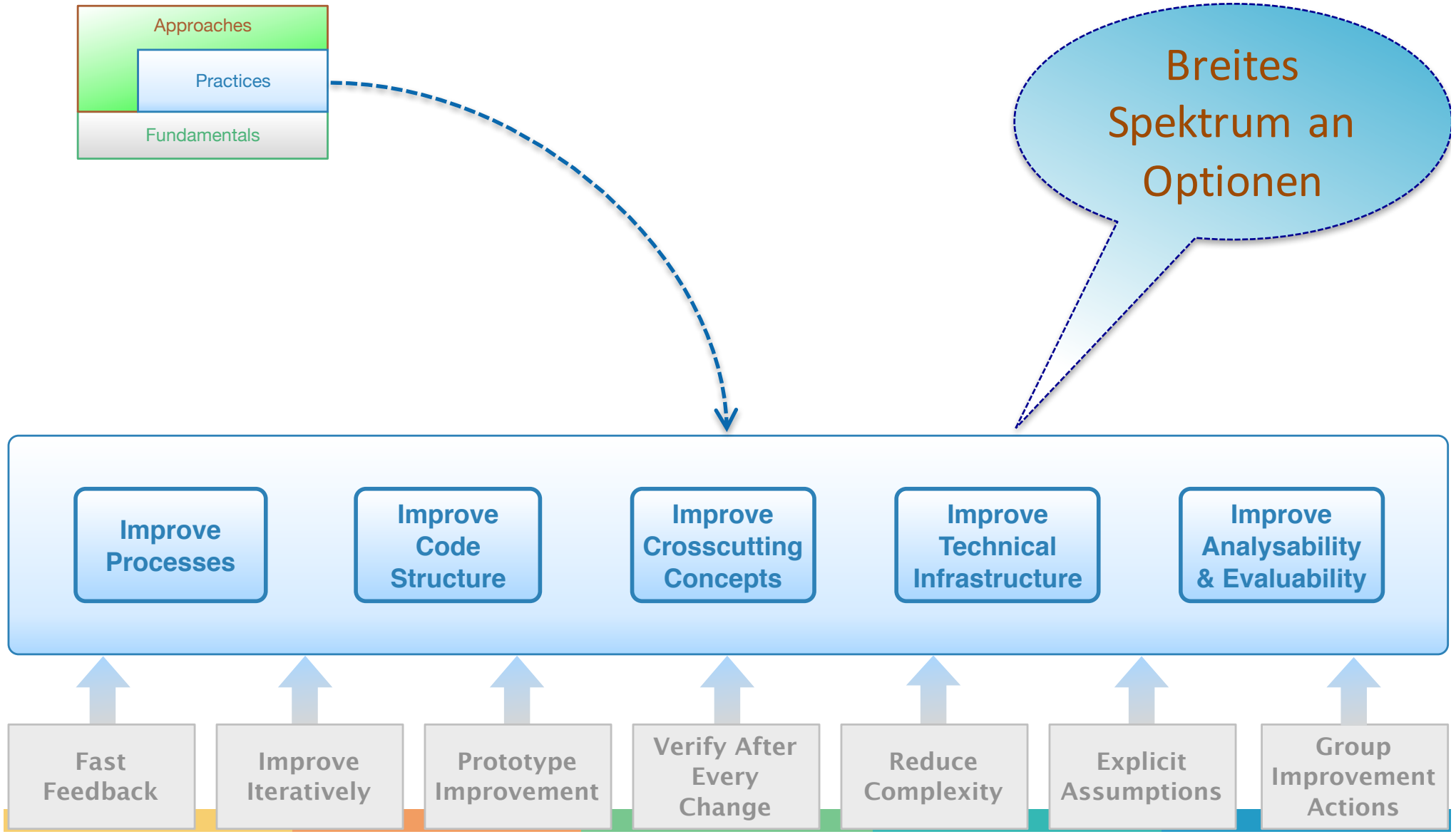
leiden nicht am Problem,
leiden unter dessen Lösung
tragen (Teil-)Schuld am Problem

zweifeln Schlussfolgerungen an,
zweifeln Ihre Kompetenz an,
haben Angst vor Veränderung

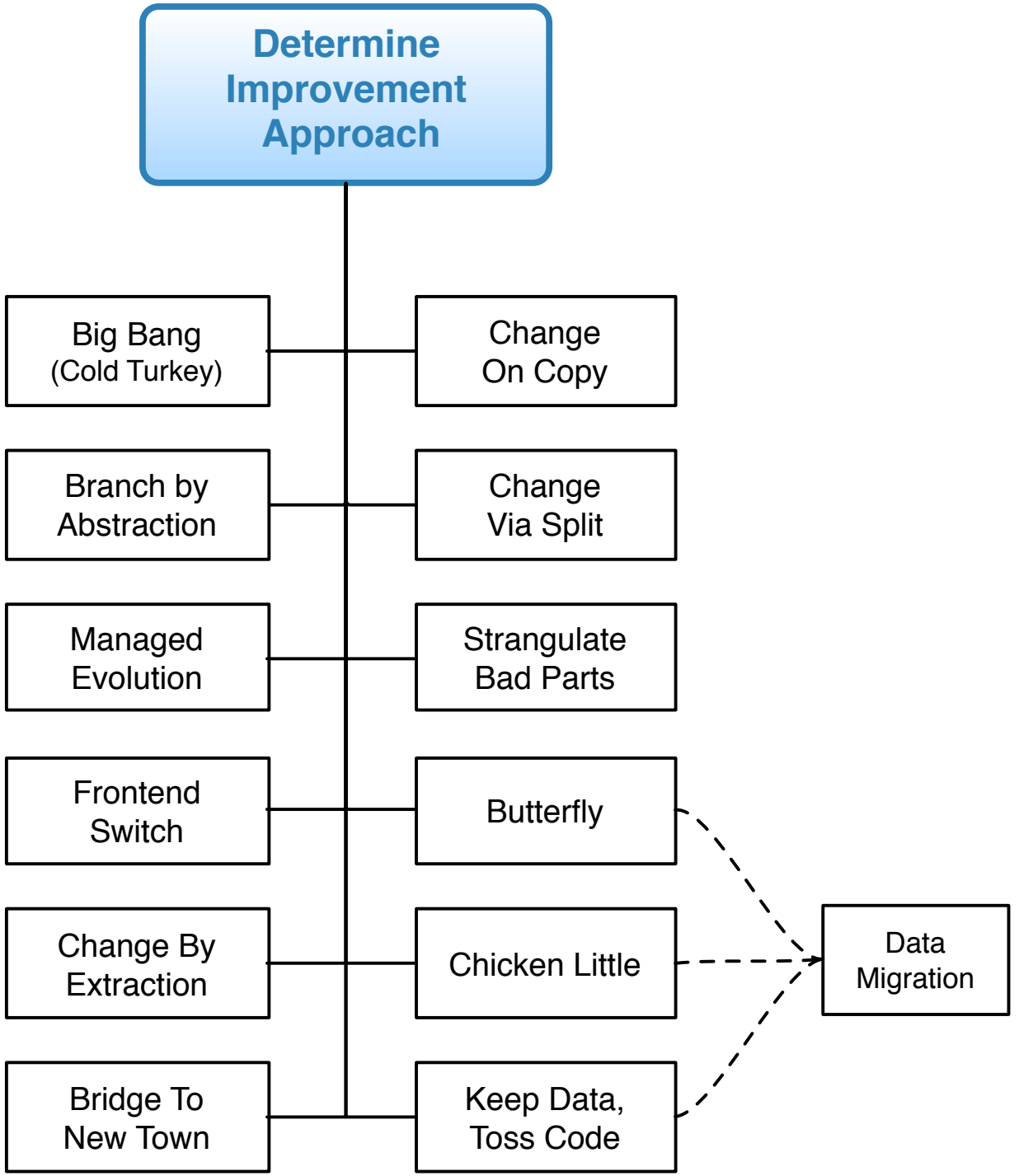
Verbesserung im Großen...



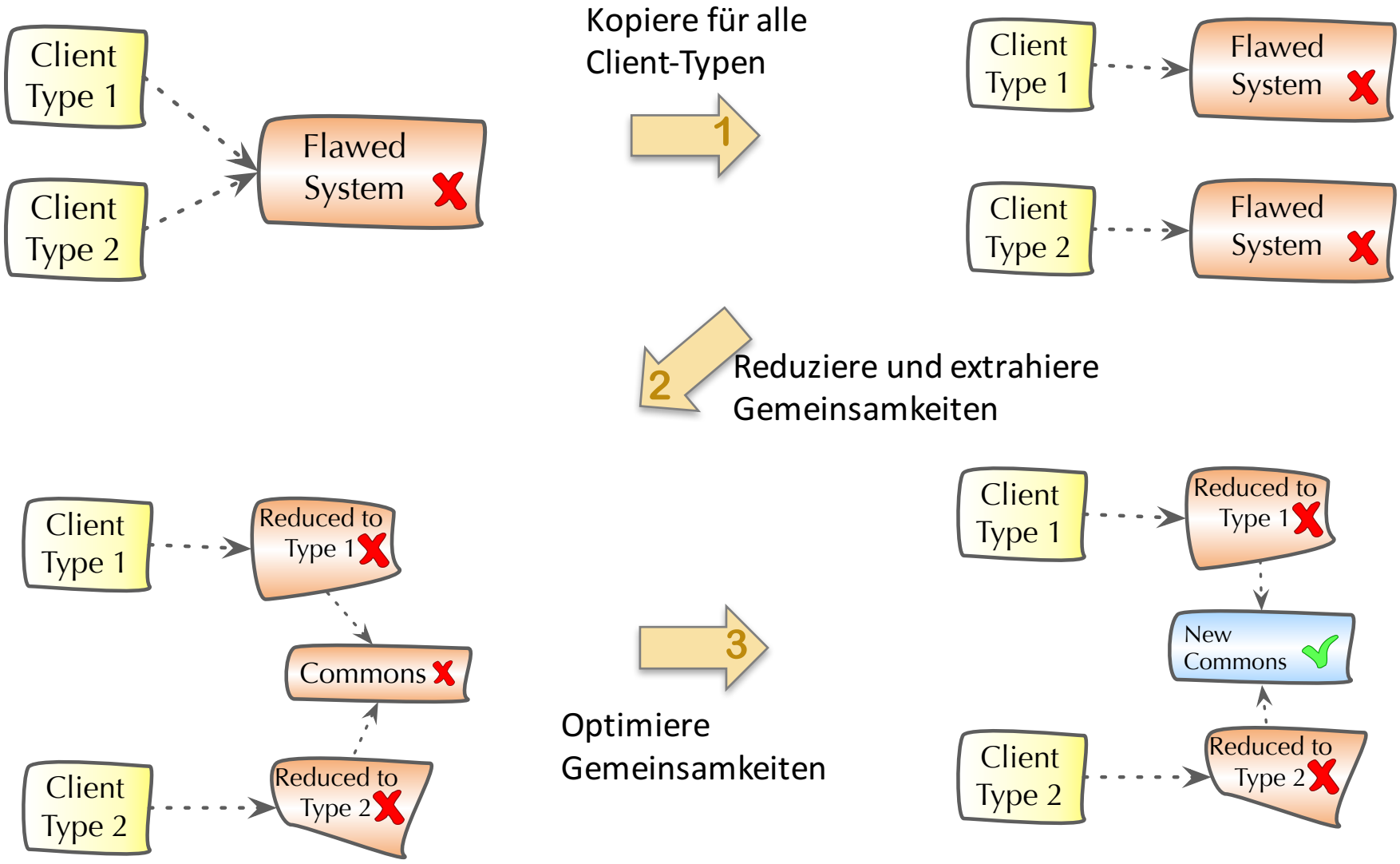
Übersicht: Praktiken



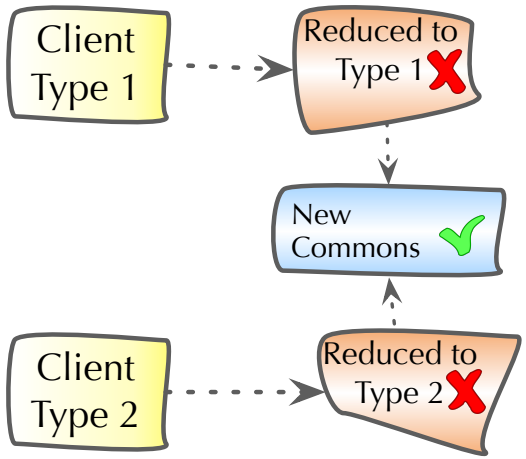
Langfristige Ansätze...



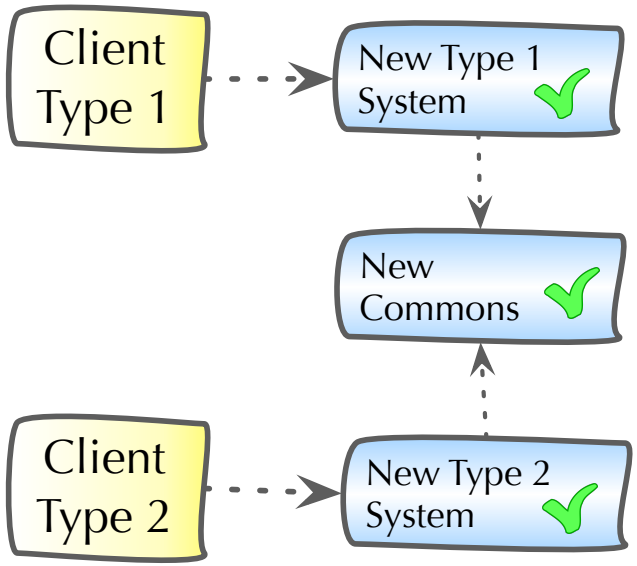
Change By Split



Change By Split (2)



Optimiere spezifische
Teilsysteme („Splits“)



Change By Split

Vorgehen

1. Identifiziere mehrere Benutzergruppen BG
2. Klone gesamtes System für jede BG
3. Reduziere für jede BG, extrahiere Gemeinsamkeiten (technische Basis)
4. Optimiere für jede BG

Voraussetzungen

- stark unterschiedliche Benutzergruppen

Risiken

- Gemeinsamkeiten schwer zu isolieren
- Mehrere Teams benötigt (1 pro Klon + Basis)

Praxis

Praktisch eingesetzt...

- Automotive: „Multimedia-Framework“
- Rail-Service „Infrastruktur“
- Mobilfunk „Billing“
- Airport-Operations „Luggage Handling“
- Systemsoftware für Maschinenbau / Lebensmittelindustrie

- 2014:
  - Europäische Bahn (Audit OnlineTicket)
  - ERP-Hersteller (Audit und Rebuild Kernsystem)

Fazit (1)

Führe

- Problem-Liste und
- Improvement-Backlog

aim42

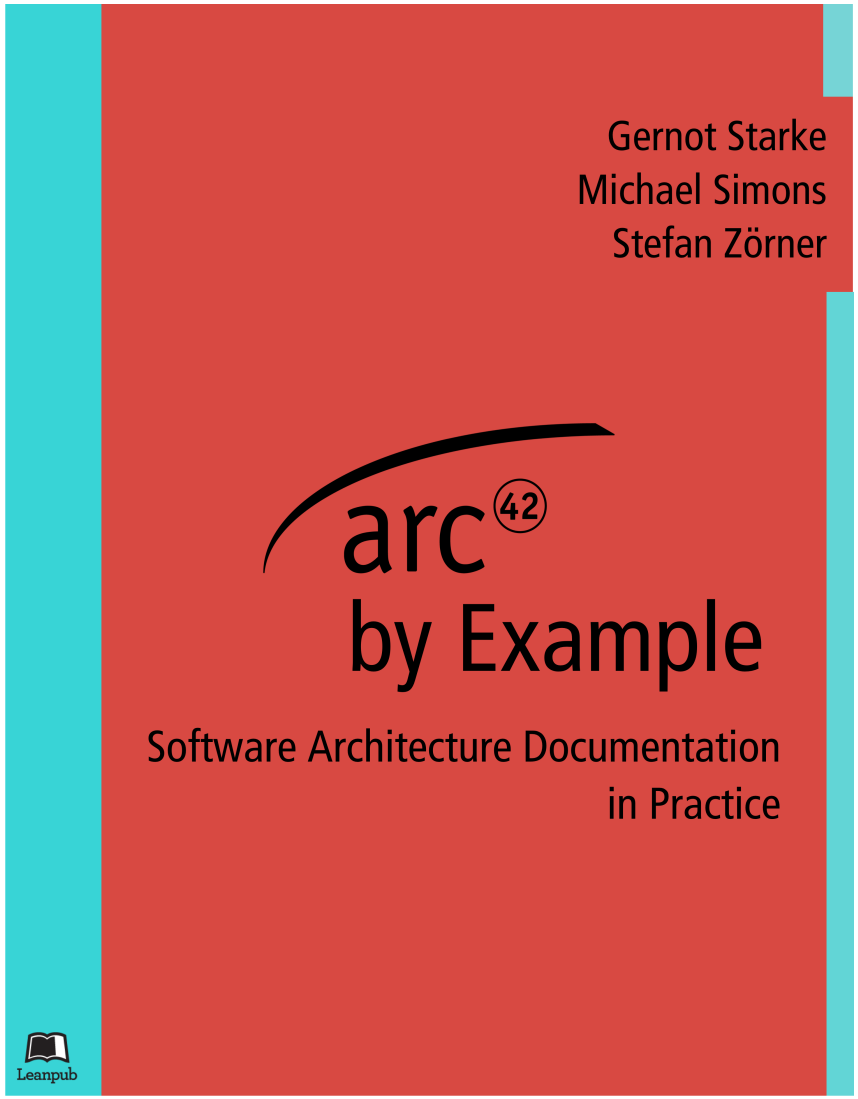
Fazit (2)

Vorsicht: Mikroskop-Falle

Fazit (3)

Vorsicht: neue Feinde...

<werbeblock>



https://leanpub.com/arc42byexample

</werbeblock>

https://www.flickr.com/photos/foto_db/16000636092

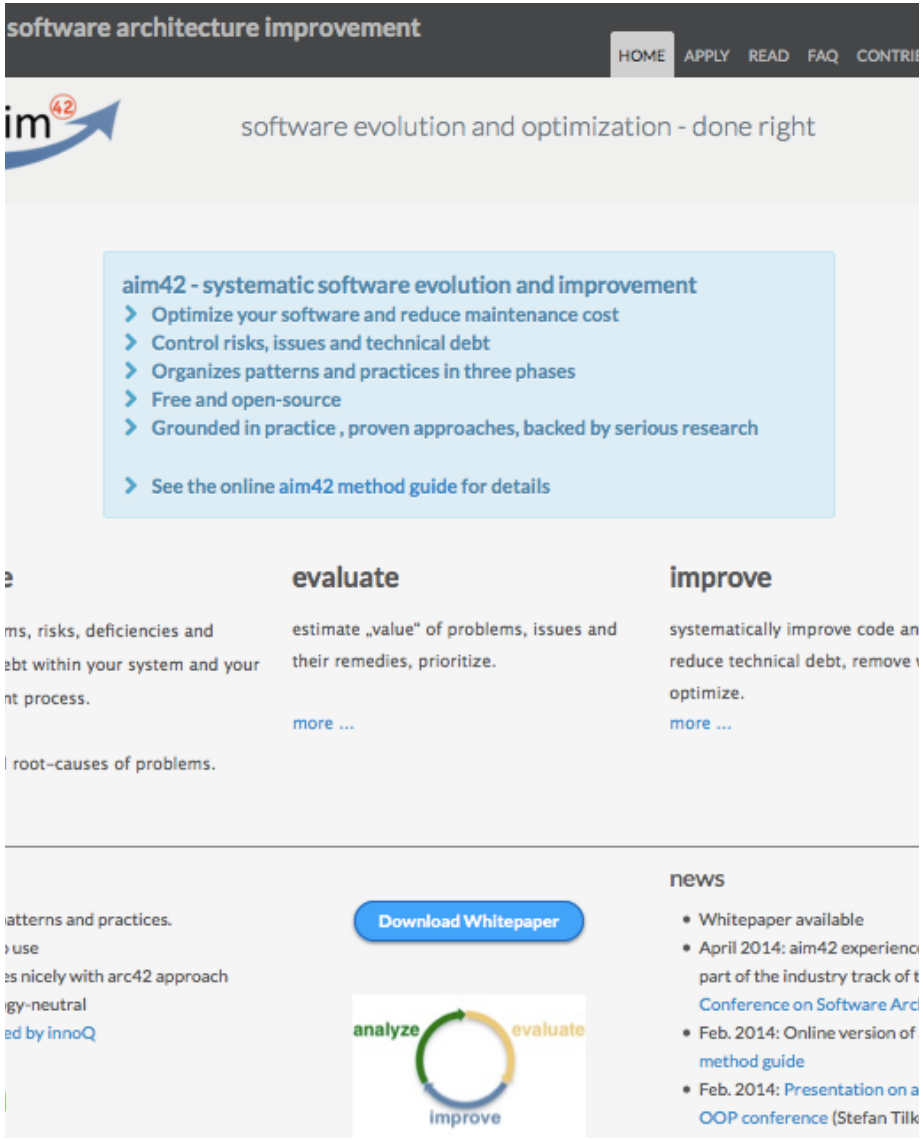
Dr. Gernot Starke

gs@gernotstarke.de

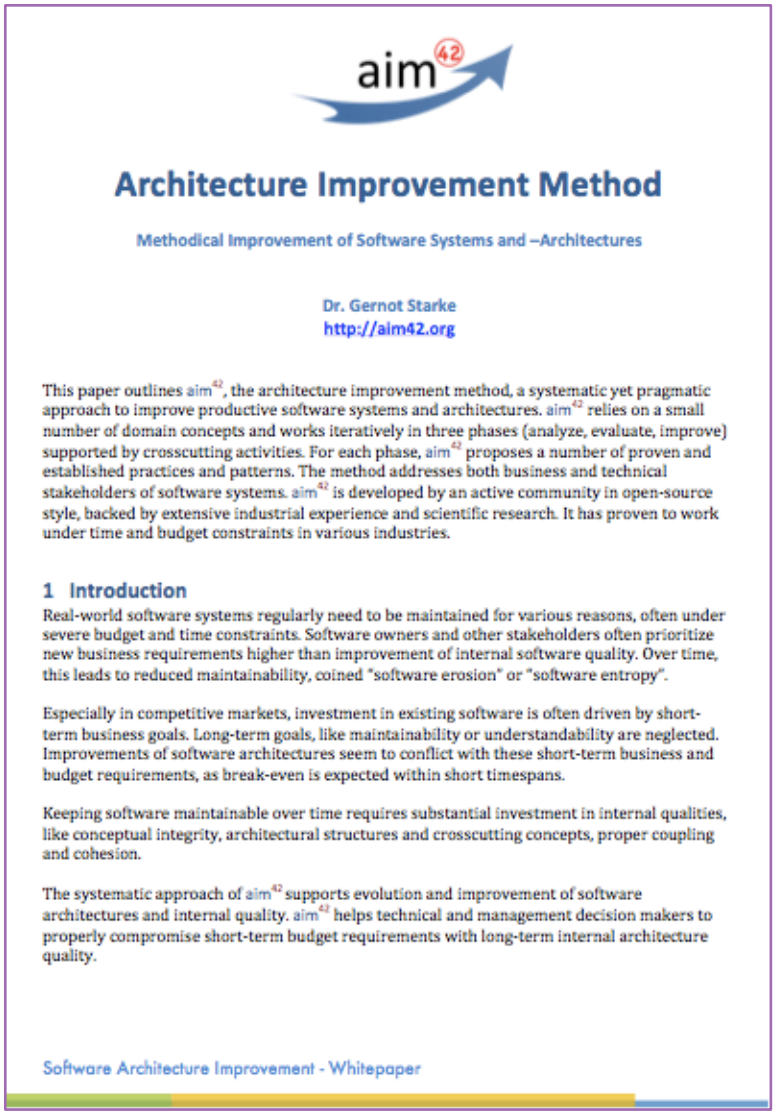
http://gernotstarke.de
http://innoq.com

Projekt

Website



Whitepaper



Architecture Improvement Method

Methodical Improvement of Software Systems and –Architectures

Dr. Gernot Starke
http://aim42.org

This paper outlines aim42, the architecture improvement method, a systematic yet pragmatic approach to improve productive software systems and architectures. aim42 relies on a small number of domain concepts and works iteratively in three phases (analyze, evaluate, improve) supported by crosscutting activities. For each phase, aim42 proposes a number of proven and established practices and patterns. The method addresses both business and technical stakeholders of software systems. aim42 is developed by an active community in open-source style, backed by extensive industrial experience and scientific research. It has proven to work under time and budget constraints in various industries.

1 Introduction

Real-world software systems regularly need to be maintained for various reasons, often under severe budget and time constraints. Software owners and other stakeholders often prioritize new business requirements higher than improvement of internal software quality. Over time, this leads to reduced maintainability, coined "software erosion" or "software entropy".

Especially in competitive markets, investment in existing software is often driven by short-term business goals. Long-term goals, like maintainability or understandability are neglected. Improvements of software architectures seem to conflict with these short-term business and budget requirements, as break-even is expected within short timespans.

Keeping software maintainable over time requires substantial investment in internal qualities, like conceptual integrity, architectural structures and crosscutting concepts, proper coupling and cohesion.

The systematic approach of aim42 supports evolution and improvement of software architectures and internal quality. aim42 helps technical and management decision makers to properly compromise short-term budget requirements with long-term internal architecture quality.

Code: github

public repo for aim42, especially the "aim42 method guide" — Edit

170 commits | **2** branches | **0** releases | **6** contributors

Your recently pushed branches:

atam (6 minutes ago) — Compare & pull request

branch: **master** — **aim42** /

Merge branch 'atam' ···

gernotstarke authored 11 hours ago — latest commit b2383582b5

graphics	added image for "view-based-understanding"	6 days ago
guide	fixing list	21 hours ago
whitepaper	improved layout of whitepaper	3 days ago
.gitignore	Merge branch 'atam'	11 hours ago
.travis.yml	Added encrypted key for repository aim42/aim42	2 months ago
readme.md	Improved image links in Readme.	2 months ago

readme.md

aim 42

Architecture Improvement Method

Code
Issues 25
Pull Requests 0
Wiki
Pulse
Graphs
Network
Settings

HTTPS clone URL

https://github.com/

You can clone with HTTPS, SSH, or Subversion.

Clone in Desktop

Download ZIP