INNOQ Technology Lunch, 08.02.2023, online

Wie löse ich knifflige Probleme in Softwaresystemen?

INNOQ

Markus Harrer
Senior Consultant

"Tools only find, people have to find out!"

Markus Harrer
Senior Consultant
Roth, Deutschland

- Softwarearchitekturentwicklung und -bewertung
- Softwaremodernisierung und -Rightsizing
- Datenanalysen in der Softwareentwicklung

aim42
http://aim42.github.io/

cards42.org

Cards for Analyzing and Reflecting on Doomed Software
https://cards42.org/

SOCREATORY
Trainer

DataCamp
Instructor

https://feststelltaste.de/

https://softwareanalytics.de

iSAQB ACCREDITED Trainer
Foundation & IMPROVE

Meine Beobachtung: Lösungsreflex

Schnelles Denken

Symptom

~95%

„Abhilfe“

L a n g s a m e s Denken

Problem

Lösung

Nach „Schnelles Denken, langsames Denken“ von Daniel Kahneman

Wir zeigen Wege für eine bessere
Zusammenarbeit jenseits der Routine auf.

Wir schätzen den Wert von:

Fragen vor **Antworten**

Beobachten vor **Bewerten**

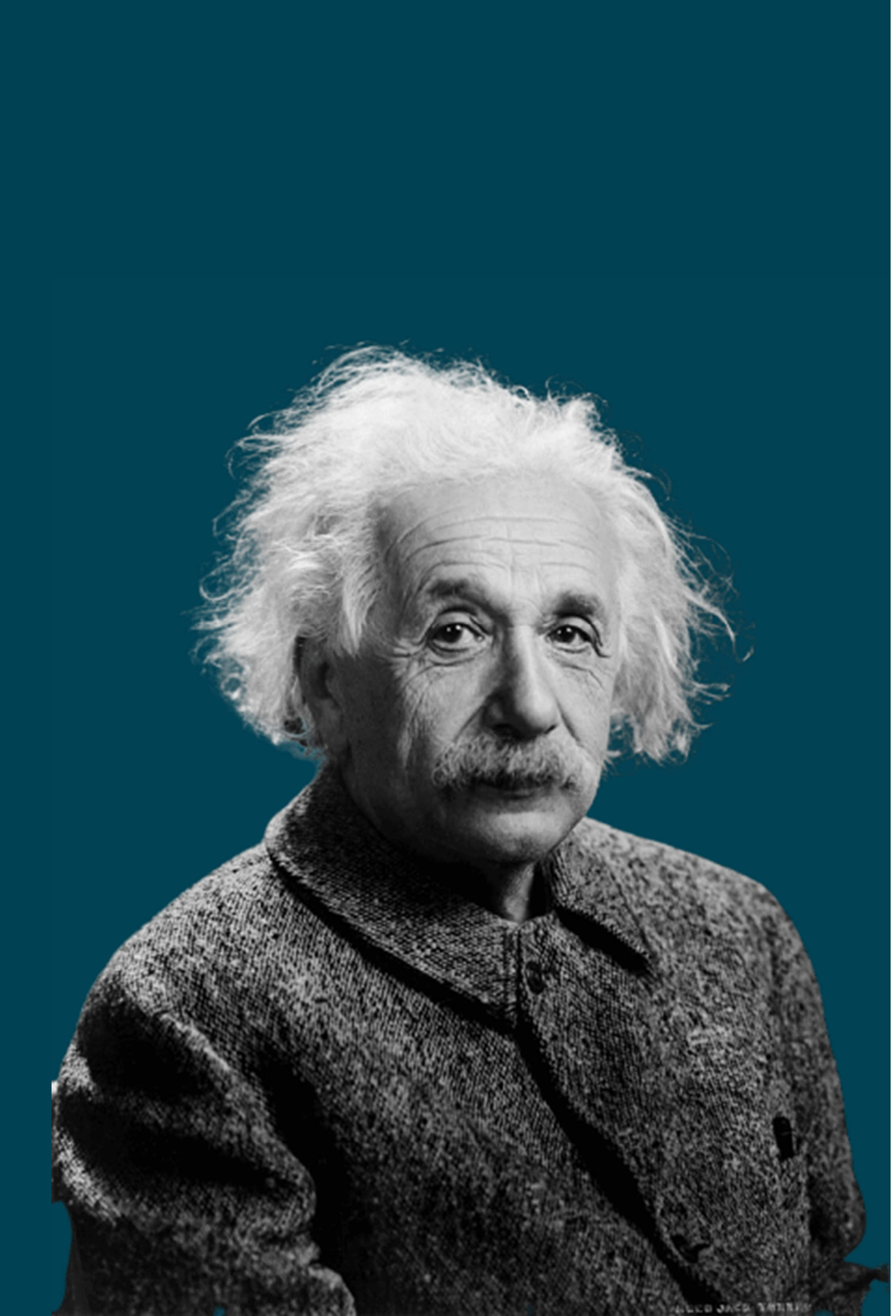
Perspektivwechsel vor **Standpunkt**

Selbstreflexion vor **Fremdkritik**

Menschen neigen zu „schnellem Denken". Dies birgt unerwünschte Effekte.
Die Aktionen der linken Seite fördern „langsames Denken". Sie steigern die Qualität
der rechten Seite und sollten daher bewusst und intensiv gelebt werden.

**Führe die Aktionen der linken Seite immer einmal mehr aus,
als Du es intuitiv für notwendig hältst!**

https://overthefence.com.de/manifesto/



Leitmotiv

"Das Problem zu erkennen ist wichtiger als die Lösung zu erkennen, denn die genaue Darstellung des Problems führt zur Lösung."

Albert Einstein

Bild: https://commons.wikimedia.org/wiki/File:Albert_Einstein_Head_cleaned.jpg

Knifflige Probleme

in Softwaresystemen

Kniffliges Problem

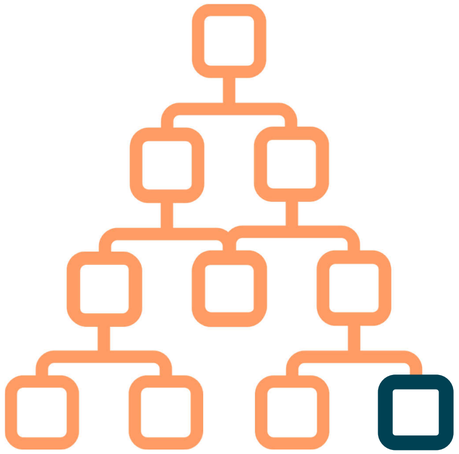
Richtiges Schlammassel

Vielfältig
Intransparent
Viele Abhängigkeiten
Wechselwirkungen

Beispiele für knifflige Probleme

Nur für den Fall, dass diese bei euch unbekannt sein sollten…

„Die Suche nach einem Vertragskunden dauert min. 5 Minuten!“



„Um ein Angebot zu berechnen, fabrizieren wir 12000 Service-Calls!“

„Nur Herr Meier (extern) kennt sich noch mit unserer Software aus!“

Der Beginn

Über Probleme reden

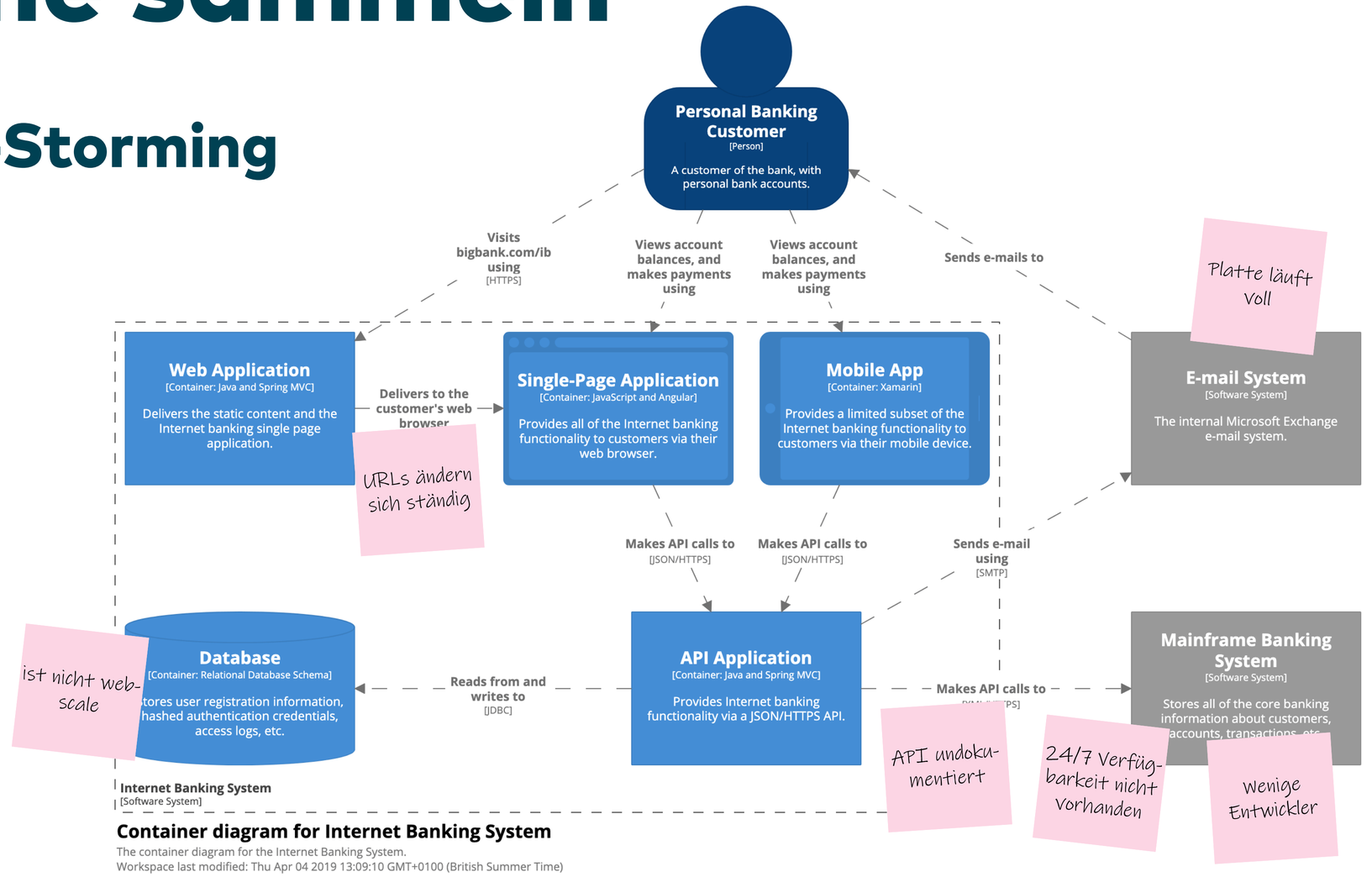
Probleme sammeln

Beispiel: The Wall of Technical Debt

Blog-Post: https://verraes.net/2020/01/wall-of-technical-debt/
Bild: https://twitter.com/Pelshoff/status/1220291781188866049, CC BY 4.0 Lizenz

Probleme sammeln

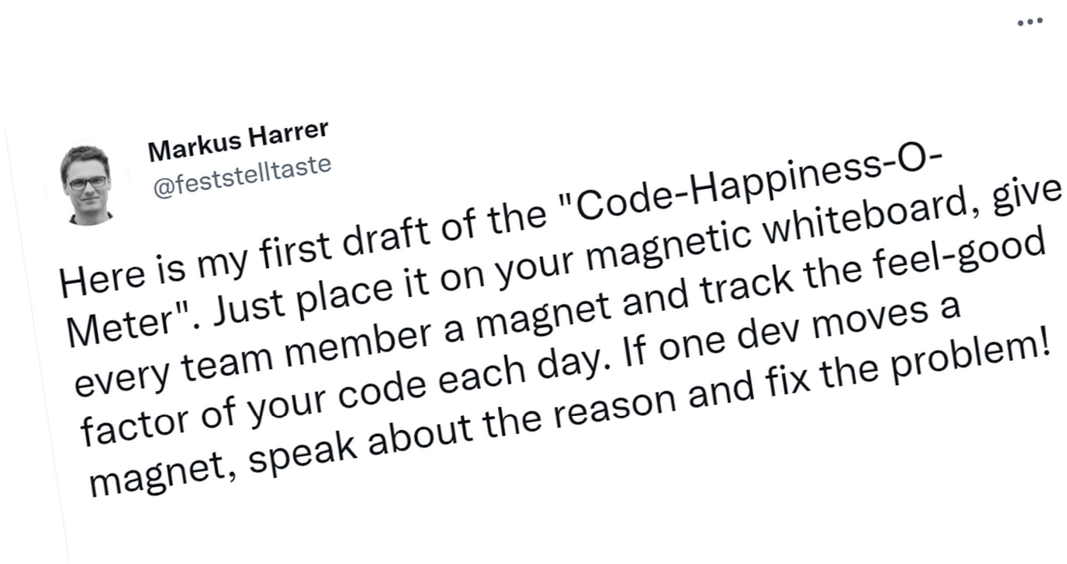
Beispiel: Risk-Storming



Blog-Post: https://riskstorming.com/
Bild: https://c4model.com, CC BY 4.0 Lizenz

Probleme sammeln

Beispiel: Code-Happiness-O-Meter



Markus Harrer
@feststelltaste

Here is my first draft of the "Code-Happiness-O-Meter". Just place it on your magnetic whiteboard, give every team member a magnet and track the feel-good factor of your code each day. If one dev moves a magnet, speak about the reason and fix the problem!

https://twitter.com/feststelltaste/status/961639365045817344

Ergebnisse einer Problemsammlung

Problem

Was nun?

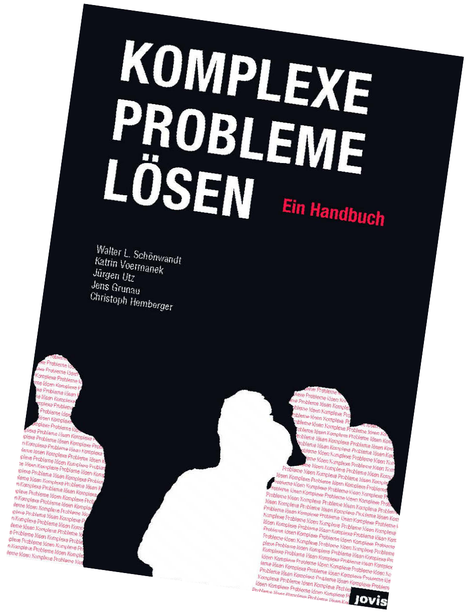
Überblick der Punkte

Reihenfolge ist irrelevant!

Wiederholungen immer und immer wieder notwendig!

1. Was sind die Probleme?
2. Problem(rück|vor)verschiebung
3. Problemverständnis
4. ProblemursachEN
5. Passende Maßnahmen
6. Denkfehler
7. Richtiges Problem?

Stark inspiriert von Walter Schönwandts Buch „Knifflige Probleme lösen“



1. Punkt

Was sind die Probleme?

Ergebnisse einer Problemsammlung

Problem

Was sind die Probleme?

Kein Kubernetes

Ø 400 DB-Calls bei 1 Klick in Anwendung

Lokaler Build geht nicht

Was sind die kniffligen Probleme?

Kein Kubernetes

Als Lösung getarntes Problem

Ø 400 DB-Calls bei 1 Klick in Anwendung

Kniffliges Problem

Lokaler Build geht nicht

Einfaches Problem (Routineaufgabe)

Ergebnisse einer Problemsammlung

Problem

Ergebnisse einer Problemsammlung

Ergebnisse einer Problemsammlung

Problem

Kein Problem

Ergebnisse einer Problemsammlung

Problem

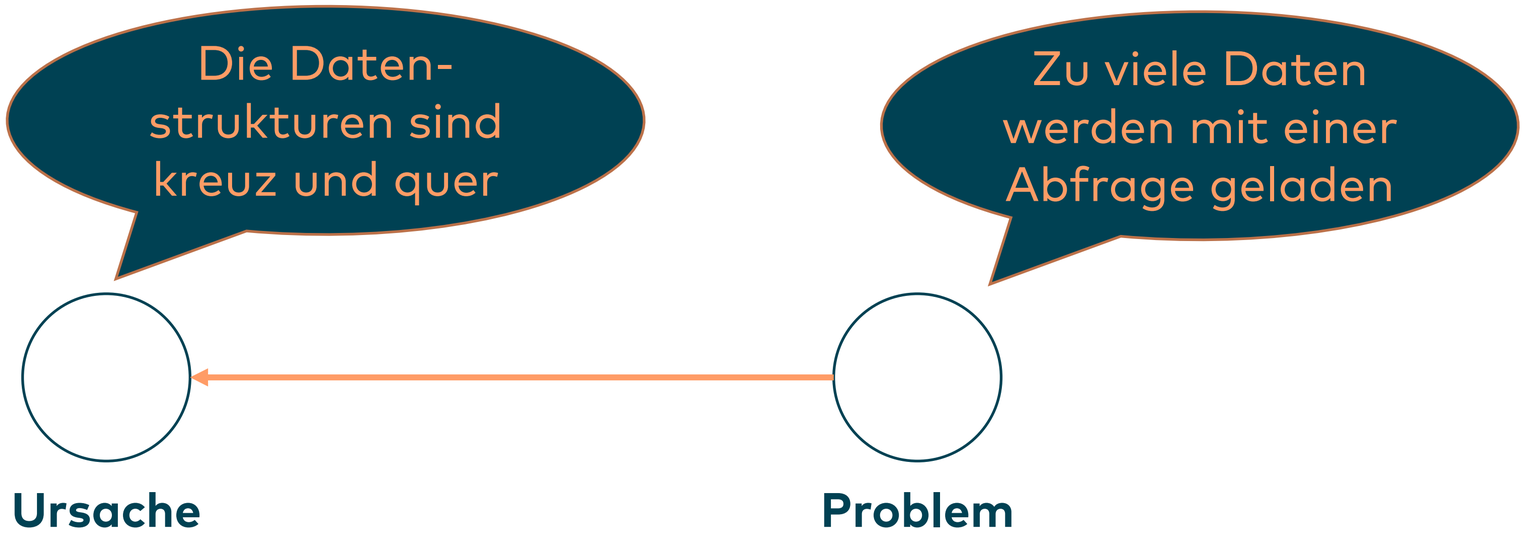
2. Punkt

Problem(rück|vor)verschiebung

Problemrückverschiebung

= Root-Cause-Analysis

→ Der Sache auf den Grund gehen

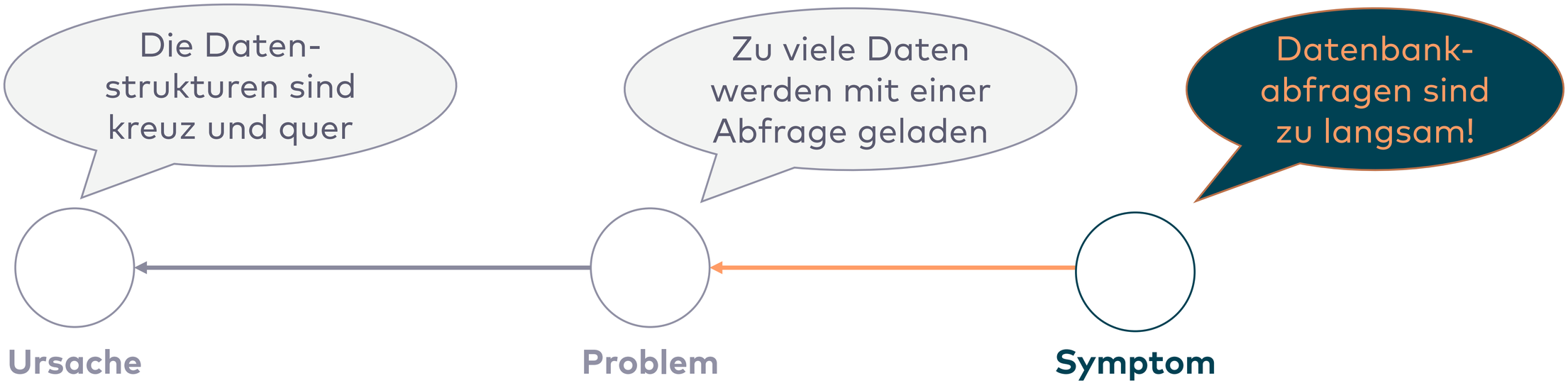


Frage: Woher kommt das Problem? (oder auch: Warum?)

Problemvorverschiebung

= „Reverse-Problem-Analysis“

→ Weitere Einstiegsstellen für die Lösung finden

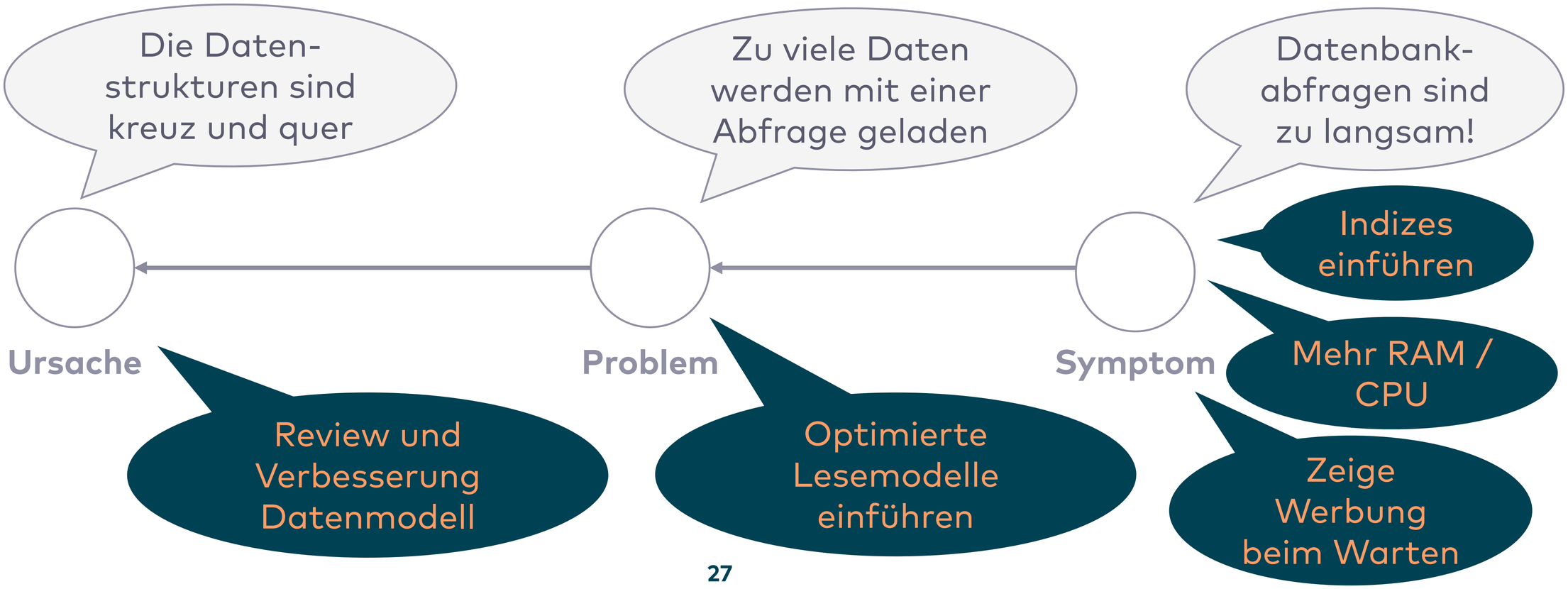


Frage: Wozu führt das Problem?

Erste Lösungsideen finden

Spielräume für die Problemlösung schaffen Räume für Ideen

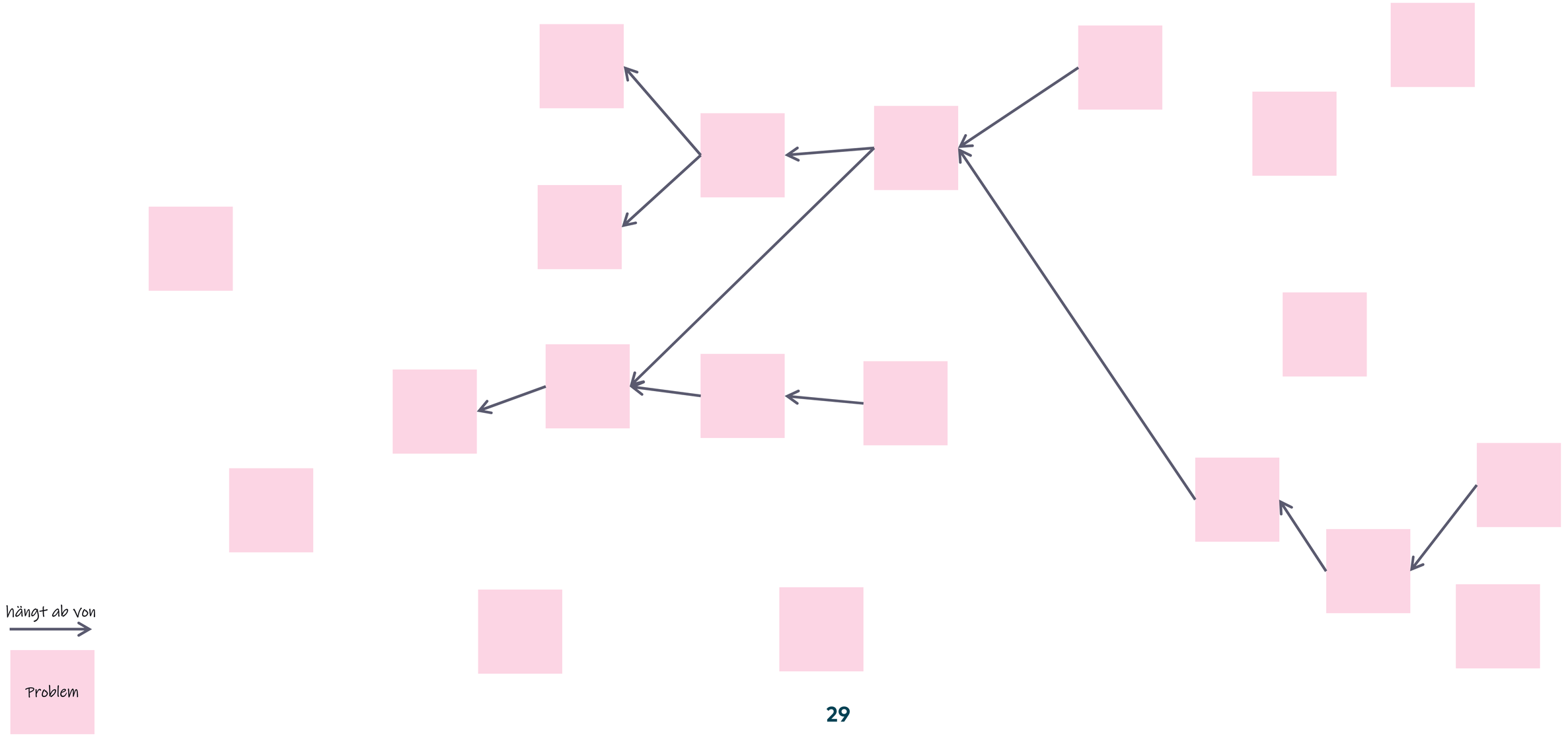
→ Viele potenzielle Dinge, über die nachgedacht werden kann



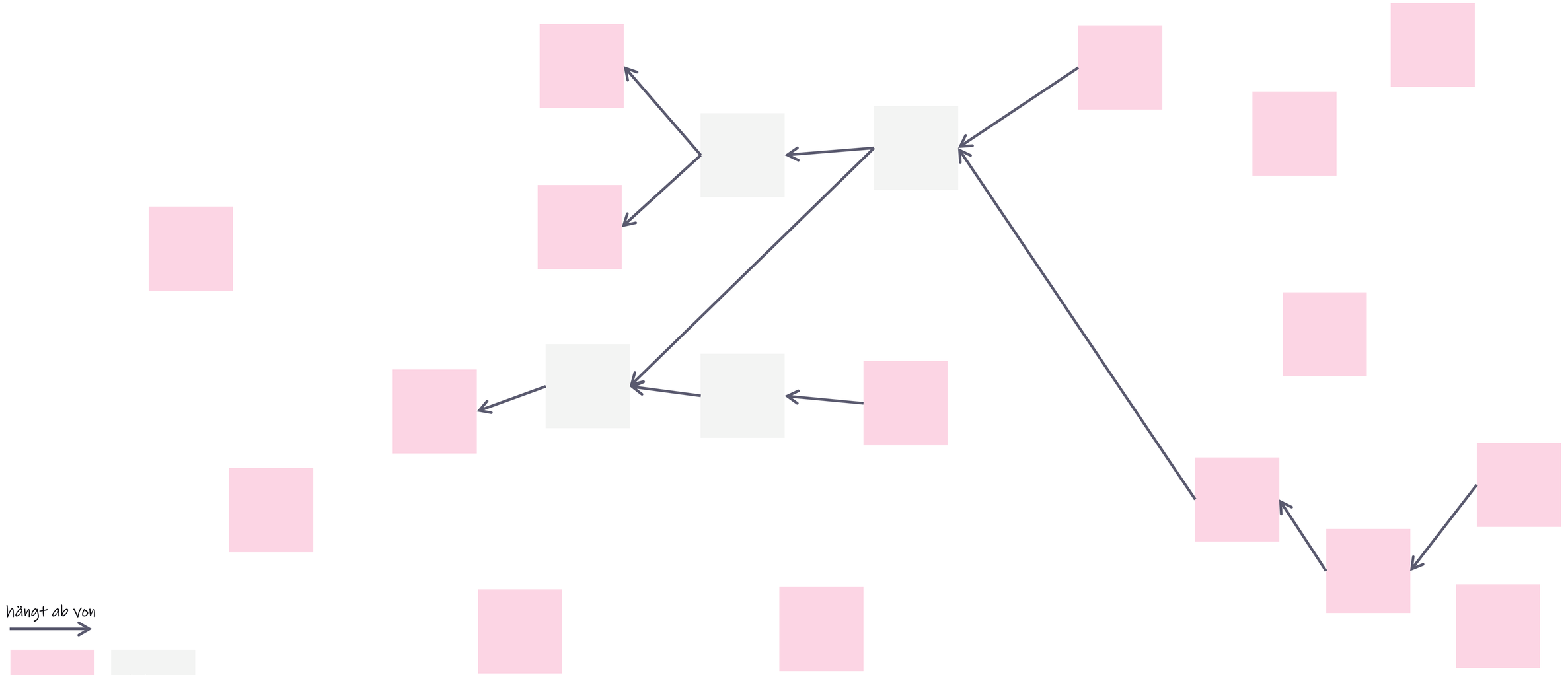
Ergebnisse einer Problemsammlung

Problem

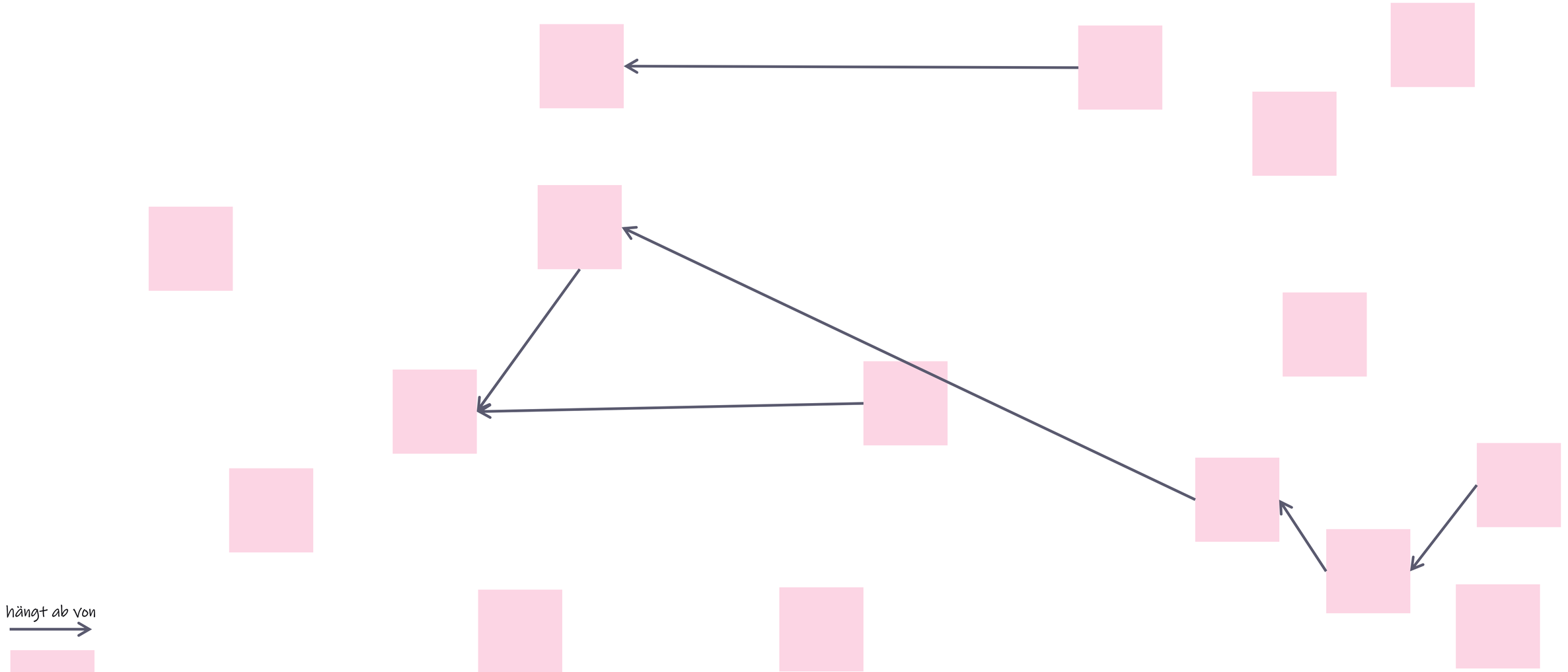
Ergebnisse einer Problemsammlung



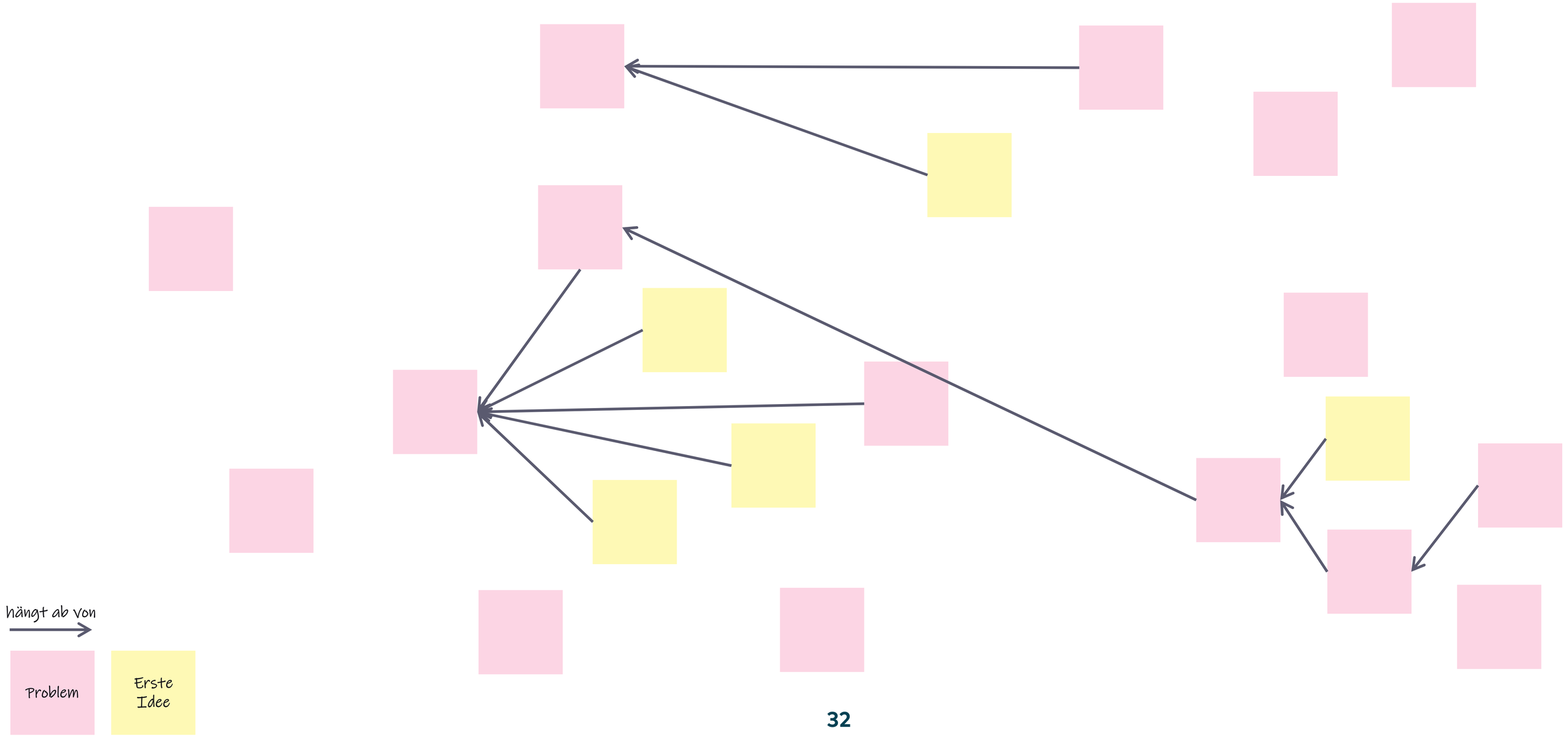
Ergebnisse einer Problemsammlung



Ergebnisse einer Problemsammlung



Ergebnisse einer Problemsammlung



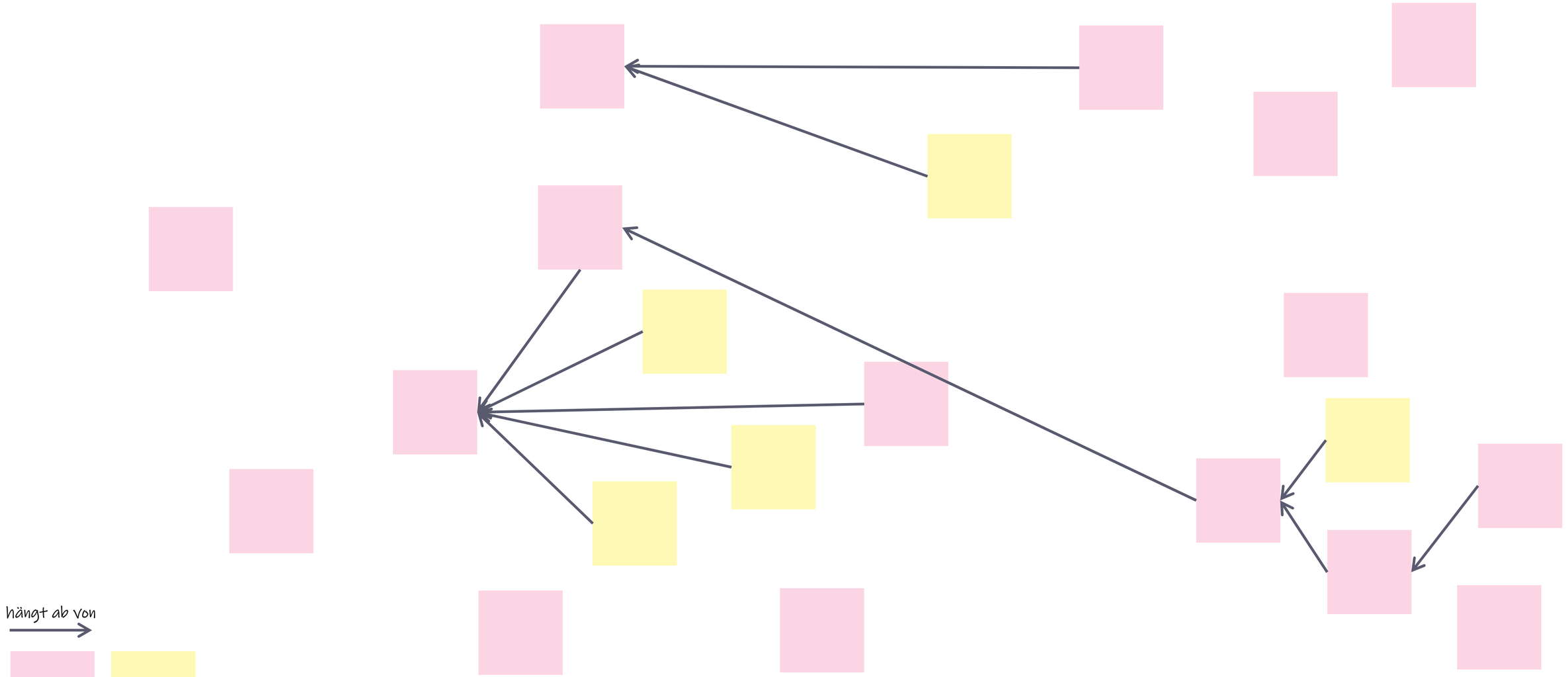
„Warum?“

„Wozu führt es?“

3. Punkt

Problemverständnis

Ergebnisse einer Problemsammlung



Ein typisches Problem

Schlechte Performance

Was können wir dagegen machen?

Typische Lösungen

Schlechte Performance

Caching

Mehr RAM / CPU

Server-less

Eine typische Situation

Schlechte Performance

= Umsetzung von Features dauert zu lange

Caching

Mehr RAM / CPU

Server-less

Resultat: Verschlimmbesserung

= Umsetzung von Features dauert zu lange

Caching

Mehr RAM / CPU

Server-less

↓ ↓ ↓

Folgen
= Quellen neuer Probleme

Daten-inkonsistenzen

Höhere Betriebs-kosten

Instabile Betriebs-umgebung

Resultat: Verschlimmbesserung

Kommt nicht so gut bei allen an…

Bei Problemen genauer werden

Antwortzeiten der Anwendung dauern zu lange?

Schlechte Performance

Umsetzung von Features dauert zu lange?

Durchsatz von Vertrags-genehmigungen zu langsam?

Entwickler tanzen zu schlecht?

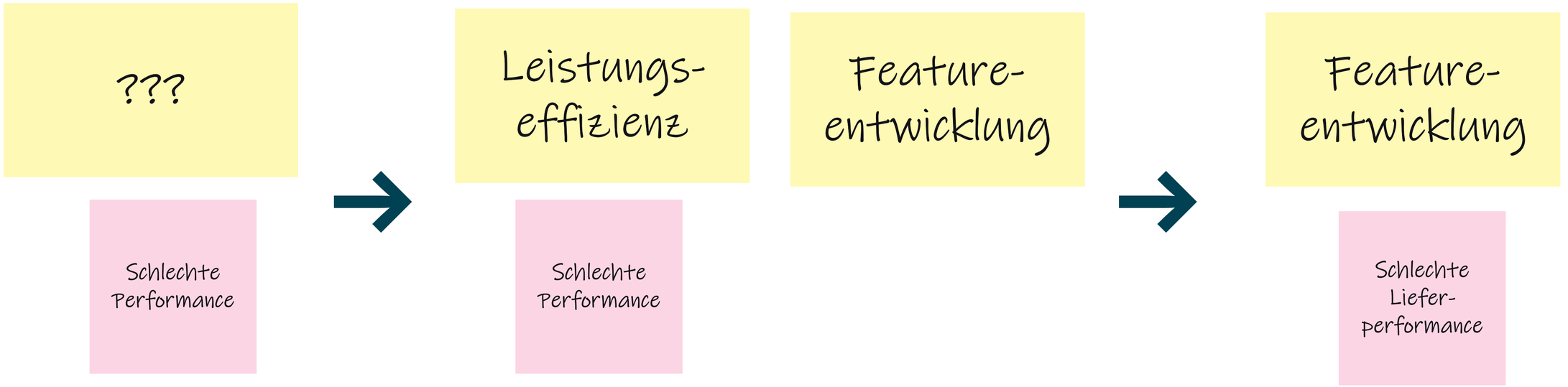
Bei Problemen genauer werden

Zu wenig Kuchen

Fehlende gegenseitige Wertschätzung

Probleme im Kontext sehen

Wörter erhalten erst in einem Kontext eine Bedeutung



→ z. B. durch Kategorisierung bzw. Clustering von Problemen

Bei Problemen konkreter werden

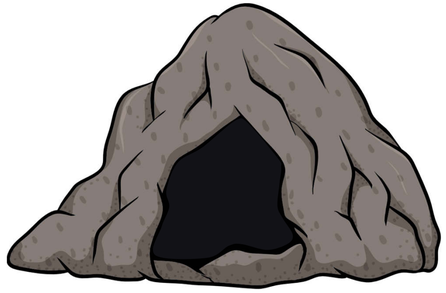
Ø 400 DB-Calls bei 1 Klick in Anwendung

>

Ø 1000 DB-Calls bei Report-Generierung

>

10000 DB-Calls bei Jahresend-berichts-generierung



Ergebnisse einer Problemsammlung

Problem

Ergebnisse einer Problemsammlung

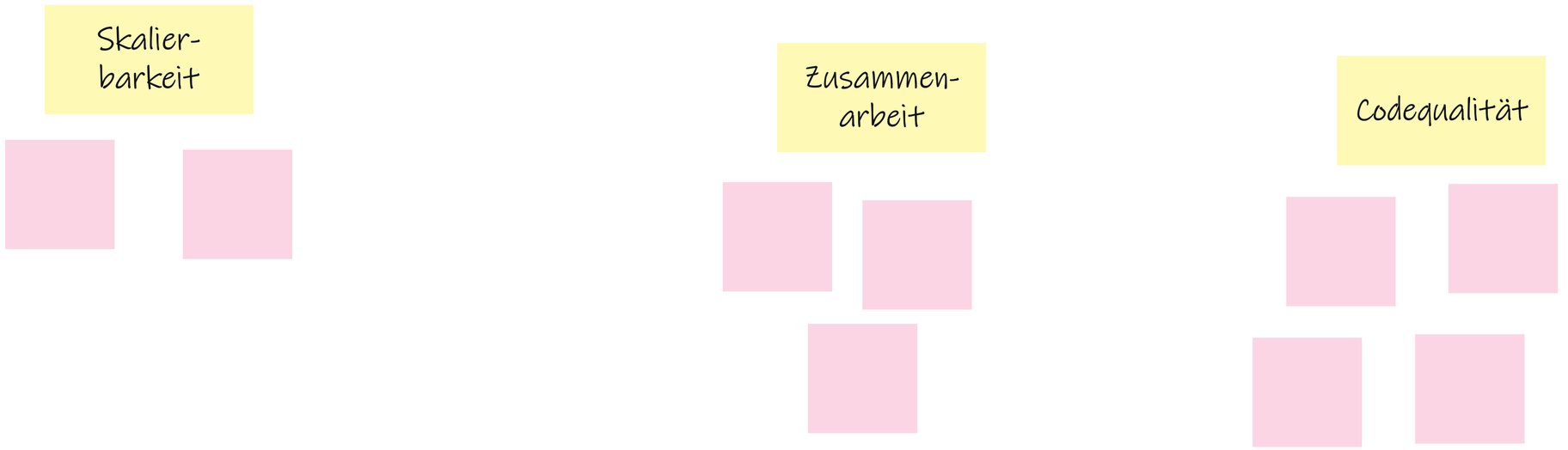
Problem

Ergebnisse einer Problemsammlung

Problem

Cluster

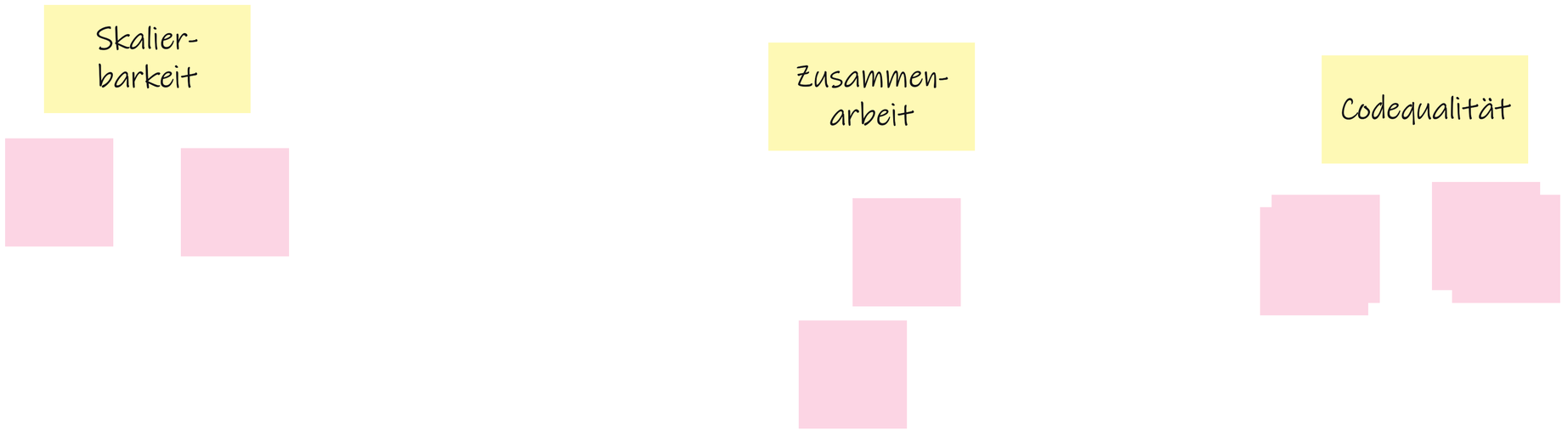
Ergebnisse einer Problemsammlung



Problem

Cluster

Ergebnisse einer Problemsammlung



Problem

Cluster

„Was meinst du damit?“

4. Punkt

ProblemursachEN

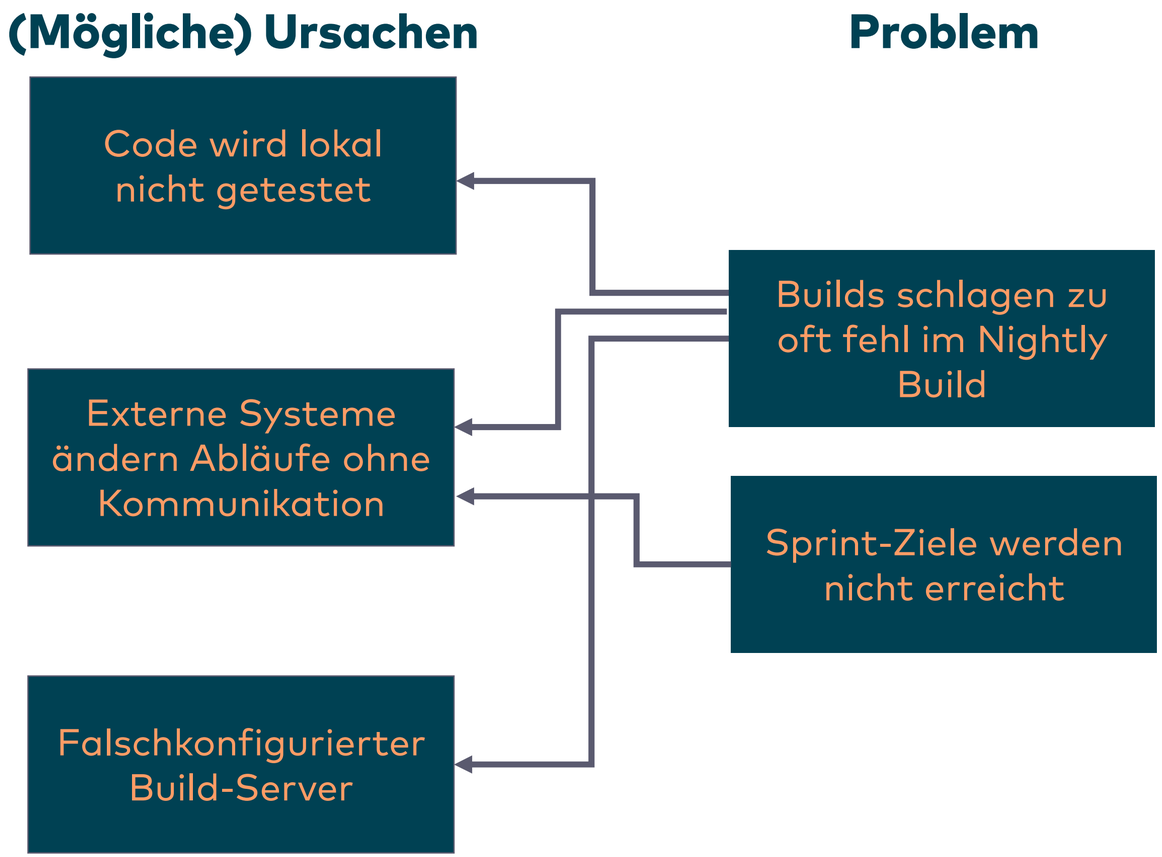
"We fail more often ***because we solve the wrong problem*** than ***because we get the wrong solution to the right problem***."

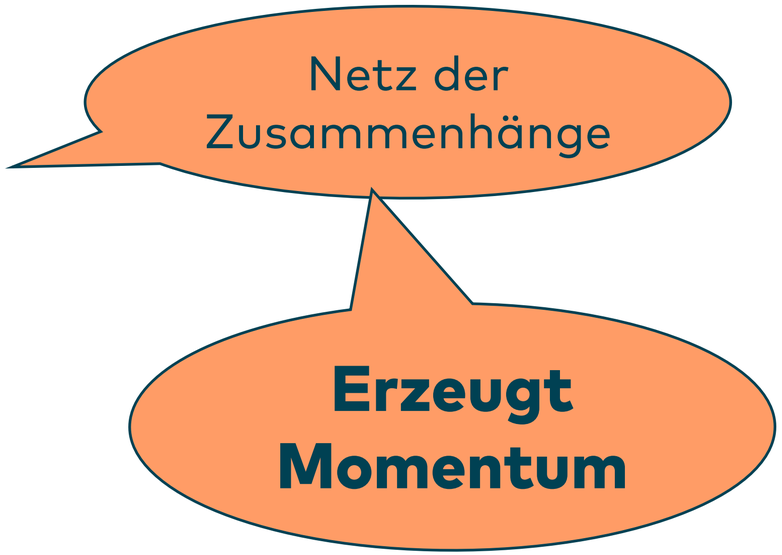
Russell L. Ackoff

Bild: https://de.wikipedia.org/wiki/Datei:Russel_Lincoln_Ackoff.jpg

Problem → UrsachEN

UrsachEN betrachten



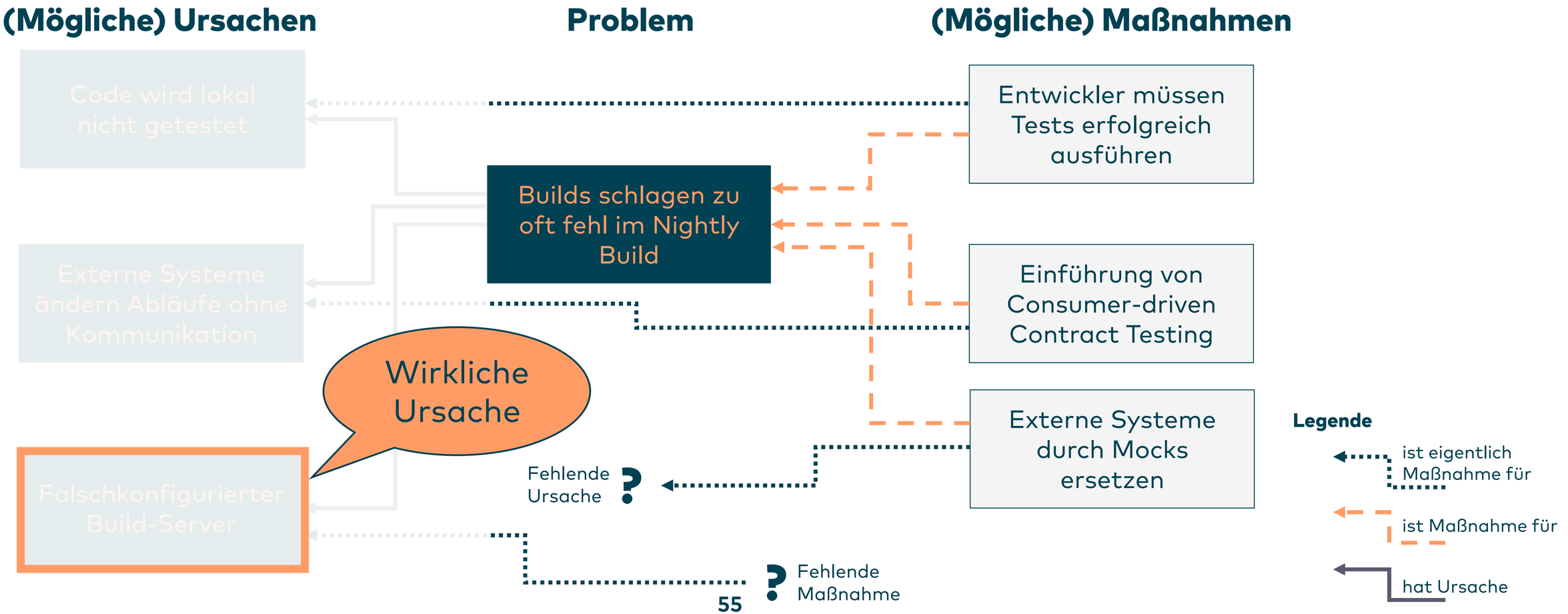


Legende

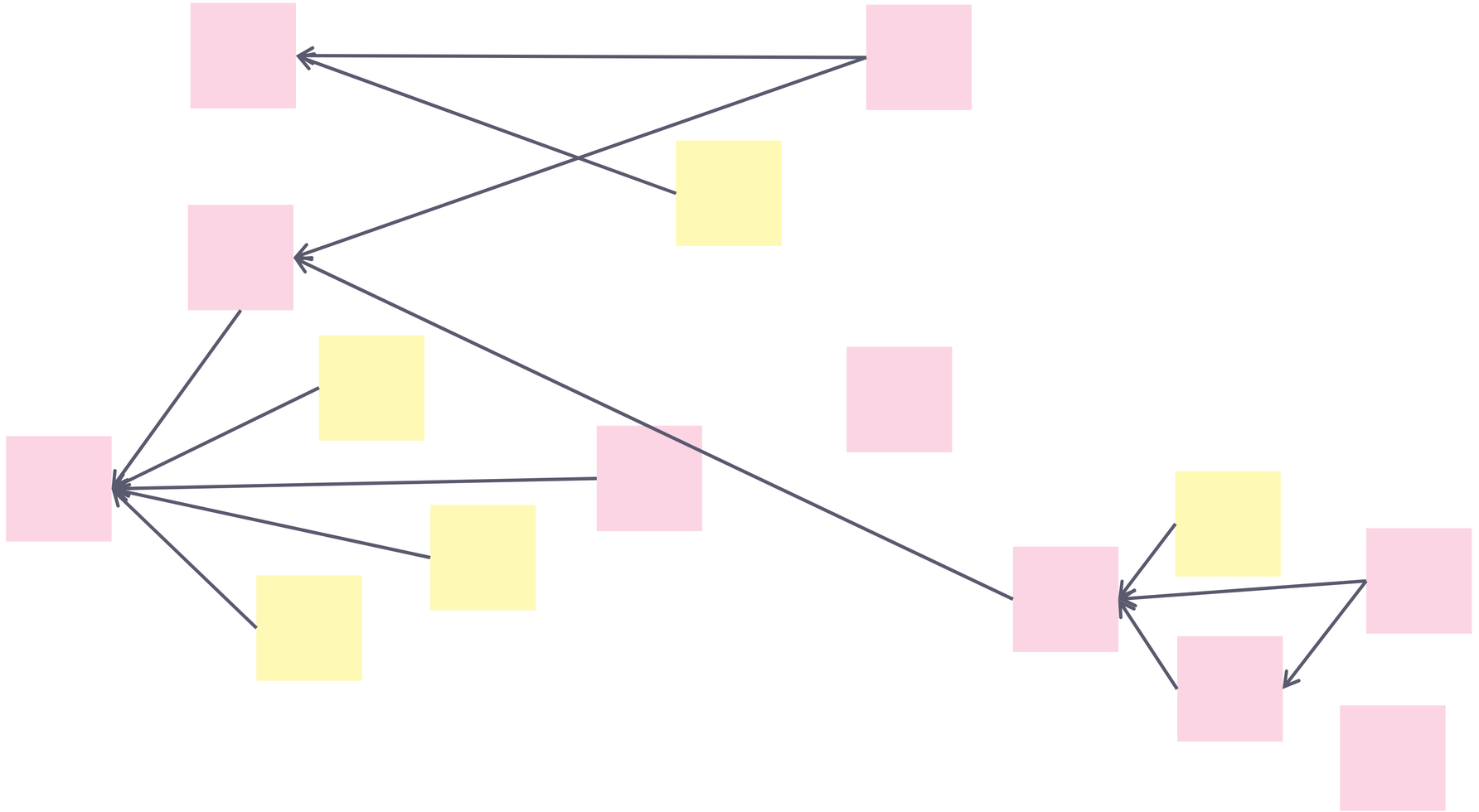
hat Ursache

Maßnahmen zu UrsachEN

Gefahr: Maßnahmen oder Ursachen unvollständig



Ergebnisse einer Problemsammlung



hängt ab von

Problem

Erste Idee

„Was noch?“

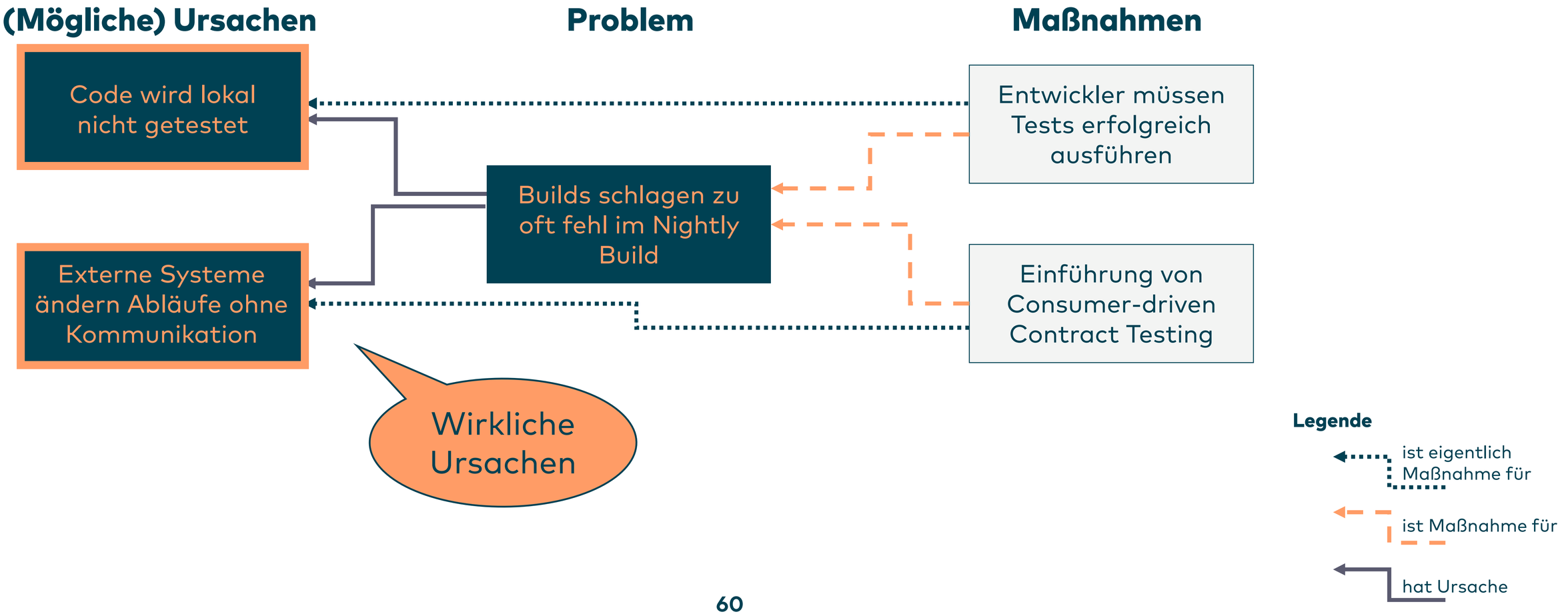
„Löst diese Maßnahme
die eigentliche Ursache?“

5. Punkt

Passende Maßnahmen finden

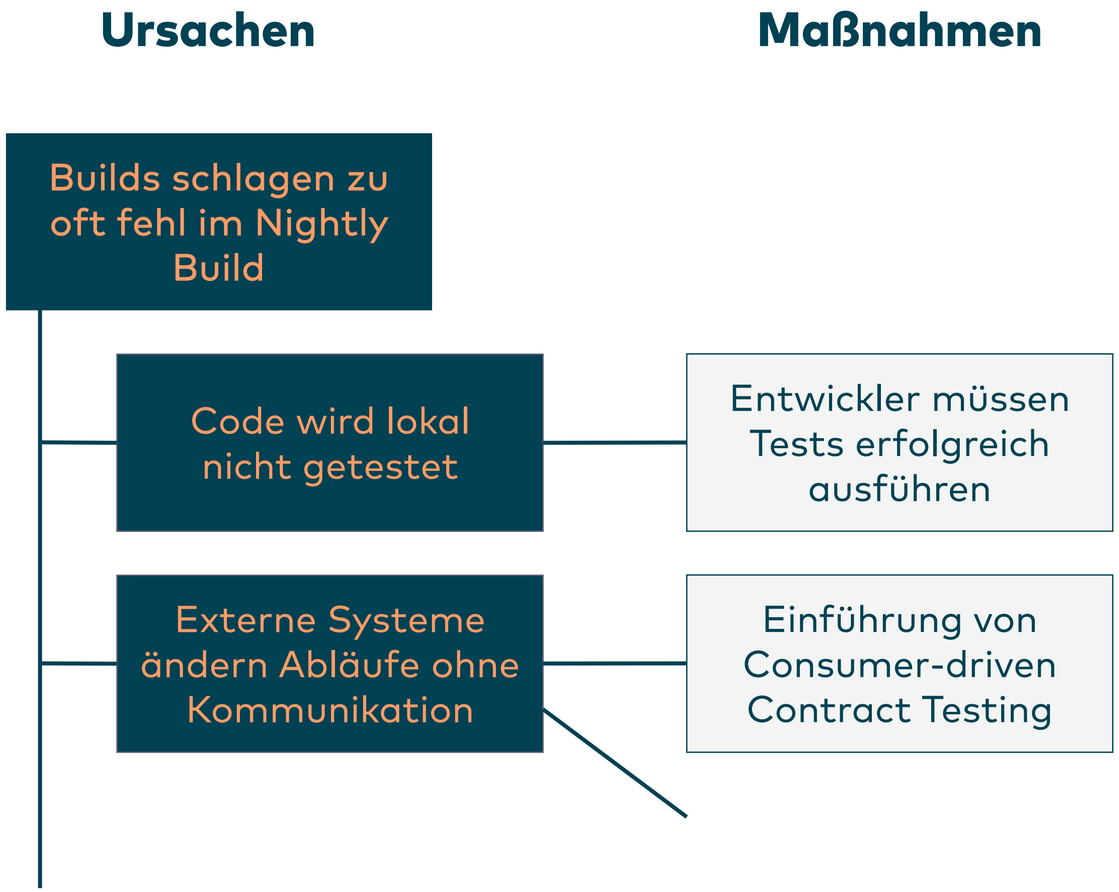
Konsequenzen von Maßnahmen

“There ain't no such thing as a free lunch!” - Unknown



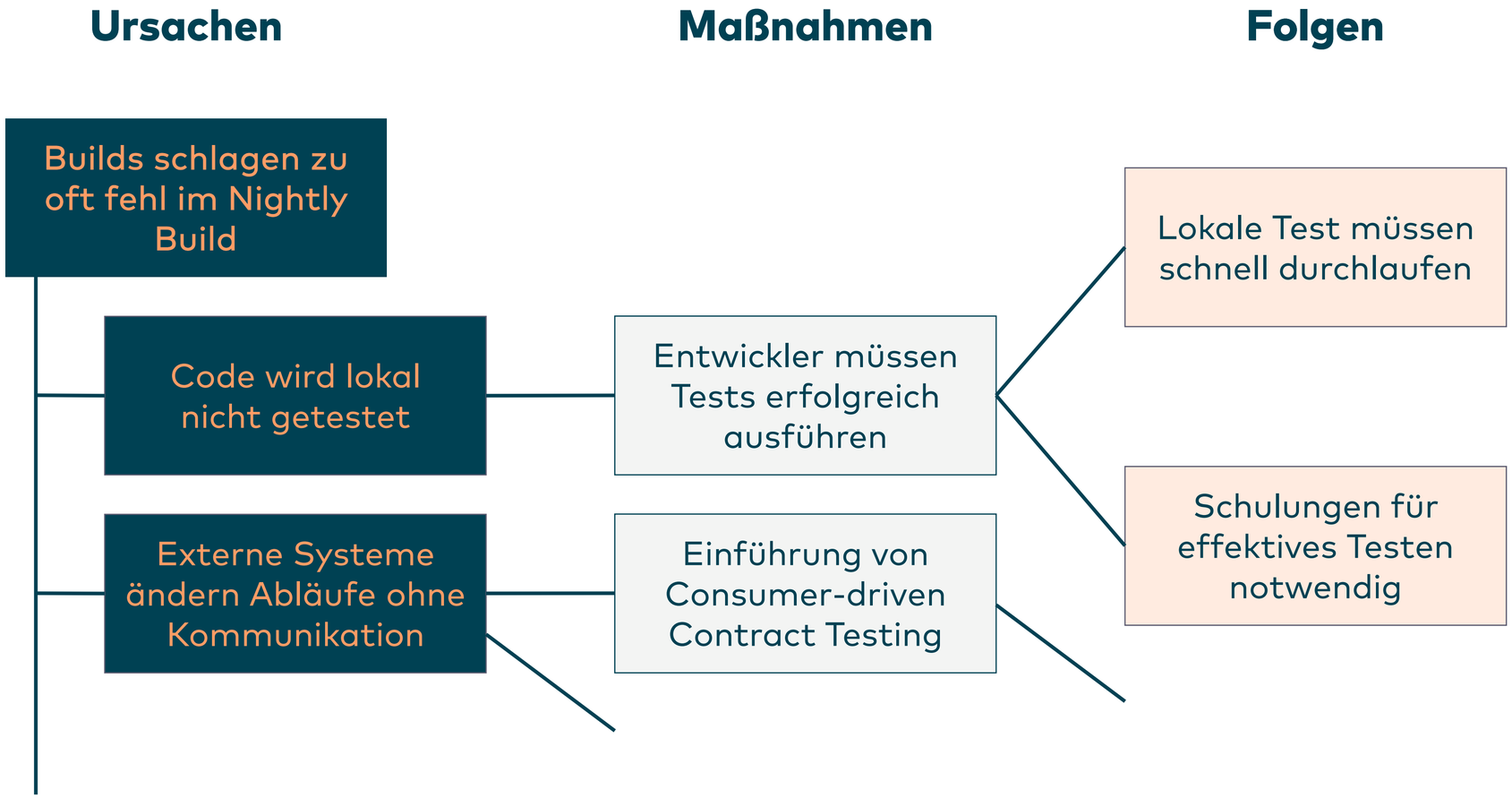
Landkarte der Lösungen

Ursachen mit Maßnahmen zusammenbringen



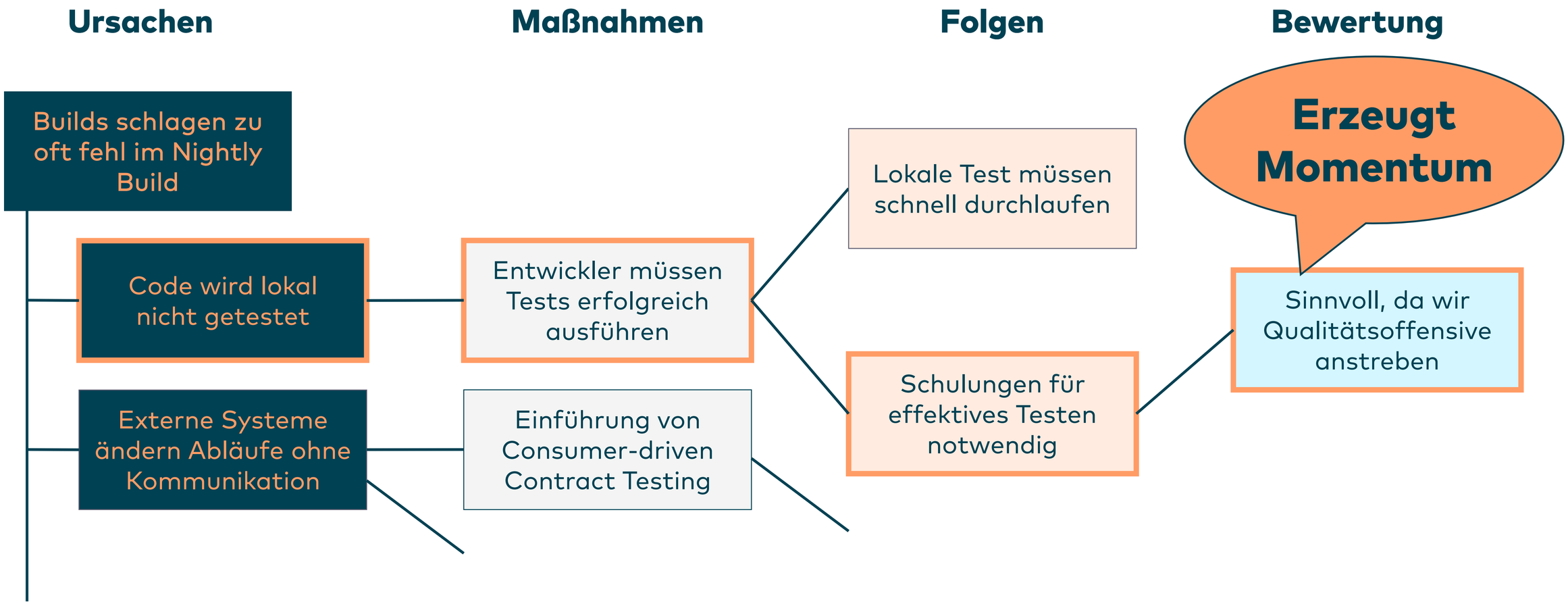
Landkarte der möglichen Lösungen

Folgen der Maßnahmen identifizieren (nichts ist umsonst!)



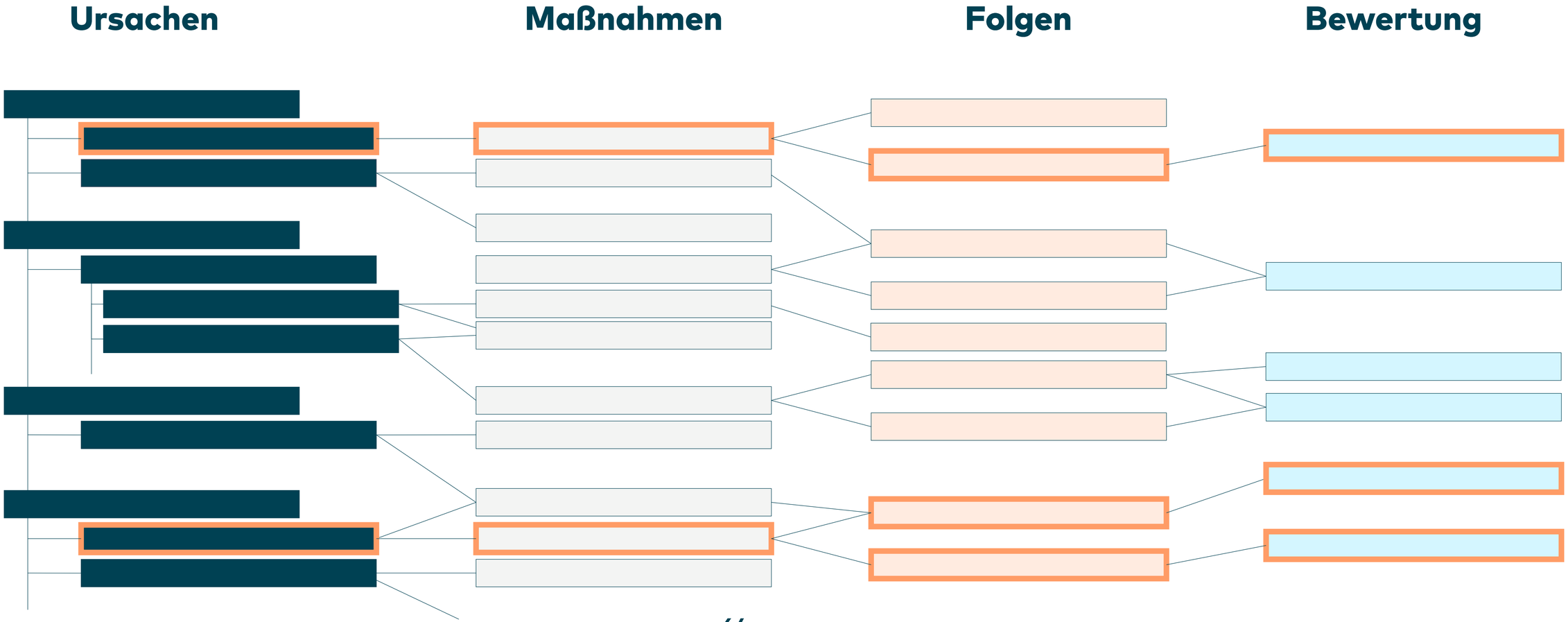
Landkarte der Lösungen

Folgen der Maßnahmen bewerten für die eigene Situation

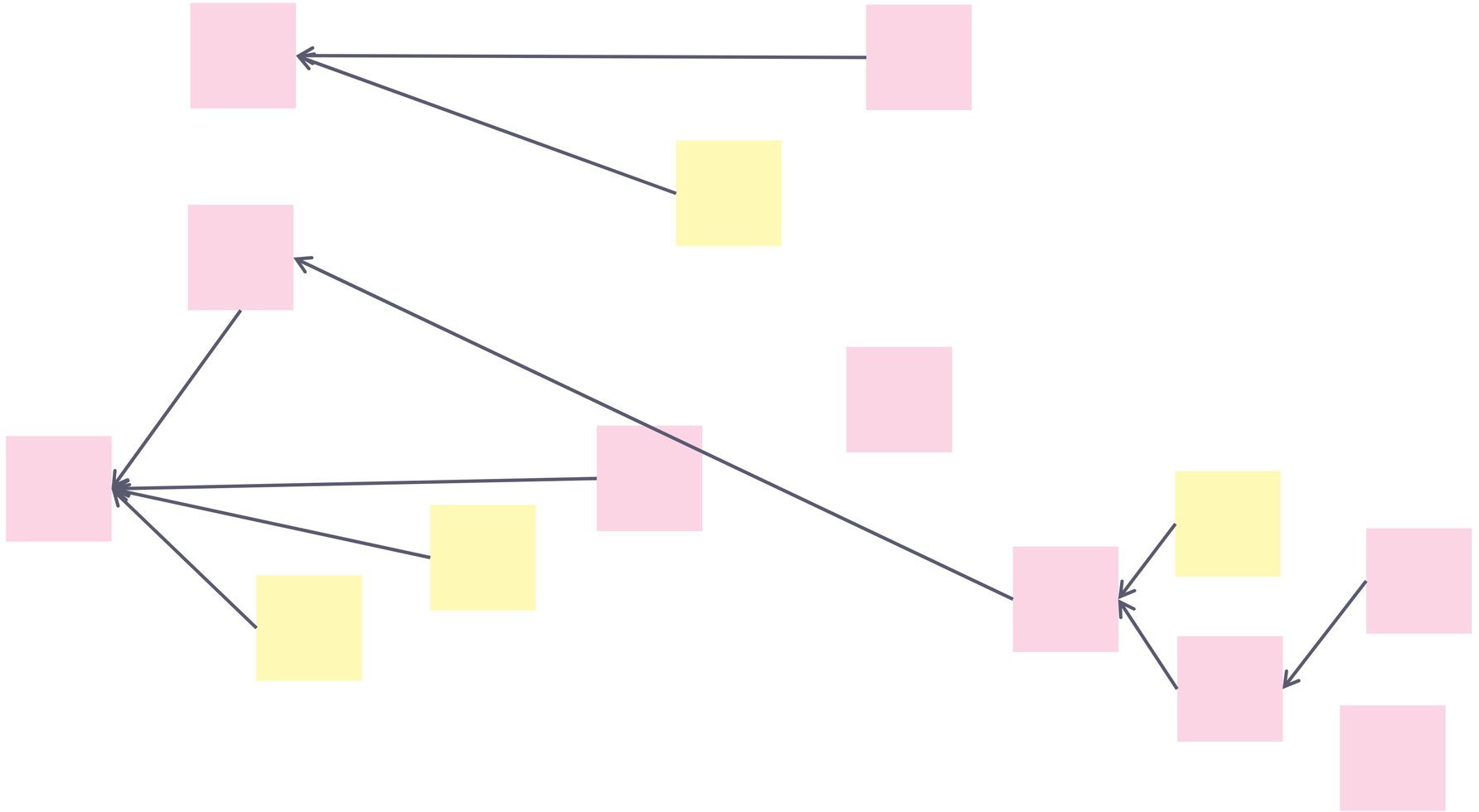


Landkarte der Lösungen

Übersicht über konkrete Maßnahmenbündel schaffen



Ergebnisse einer Problemsammlung

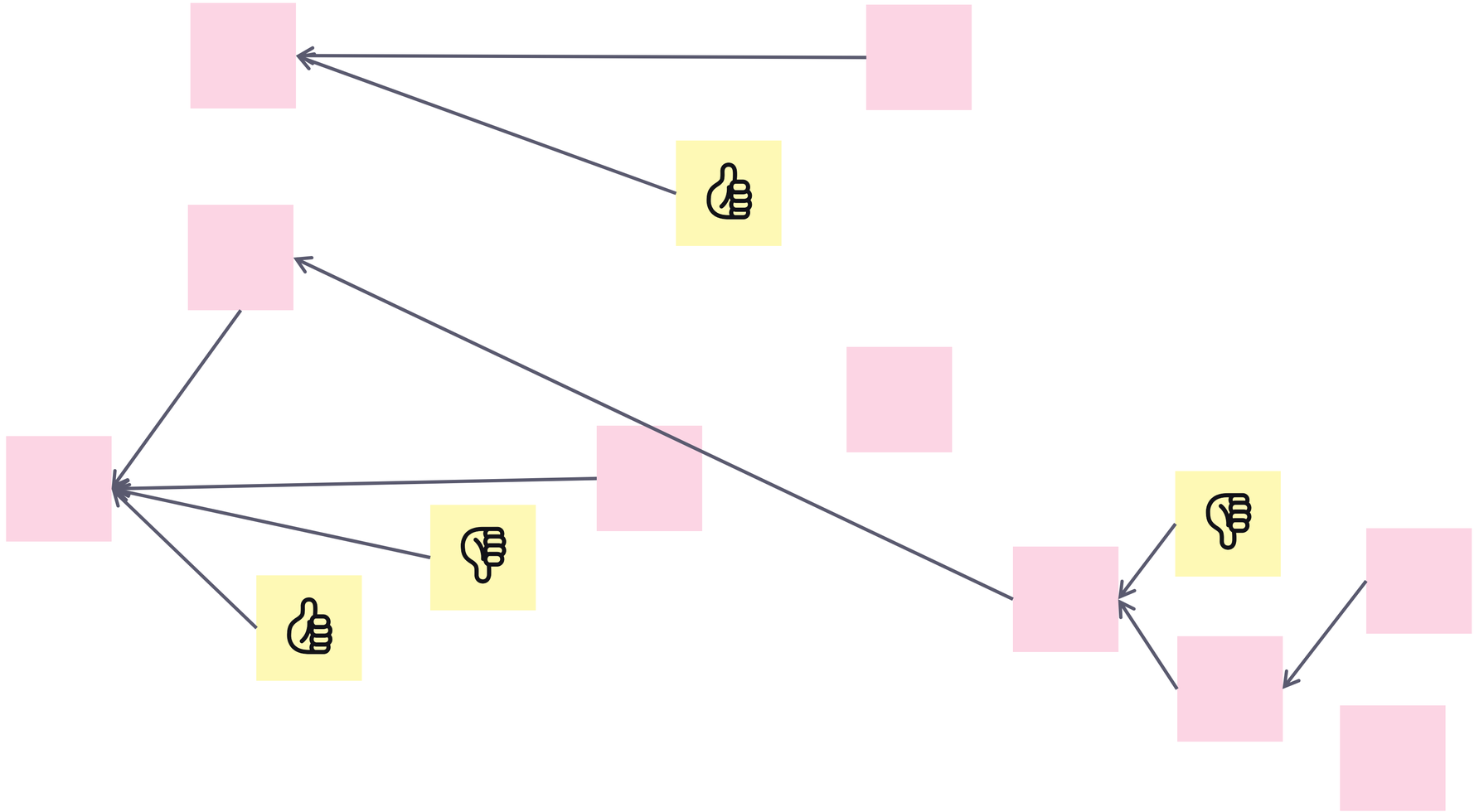


hängt ab von

Problem

Erste Idee

Ergebnisse einer Problemsammlung

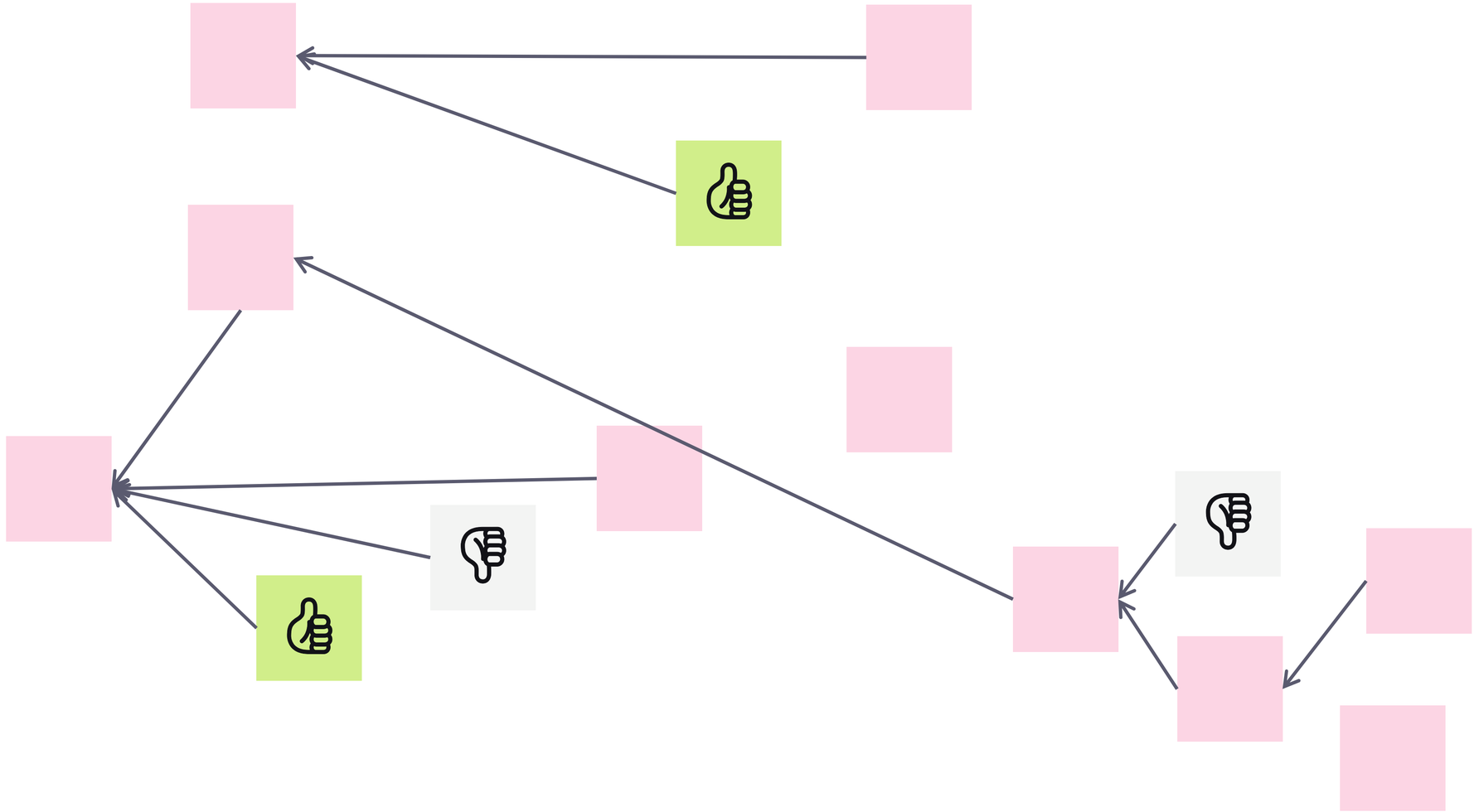


hängt ab von

Problem

Erste Idee

Ergebnisse einer Problemsammlung



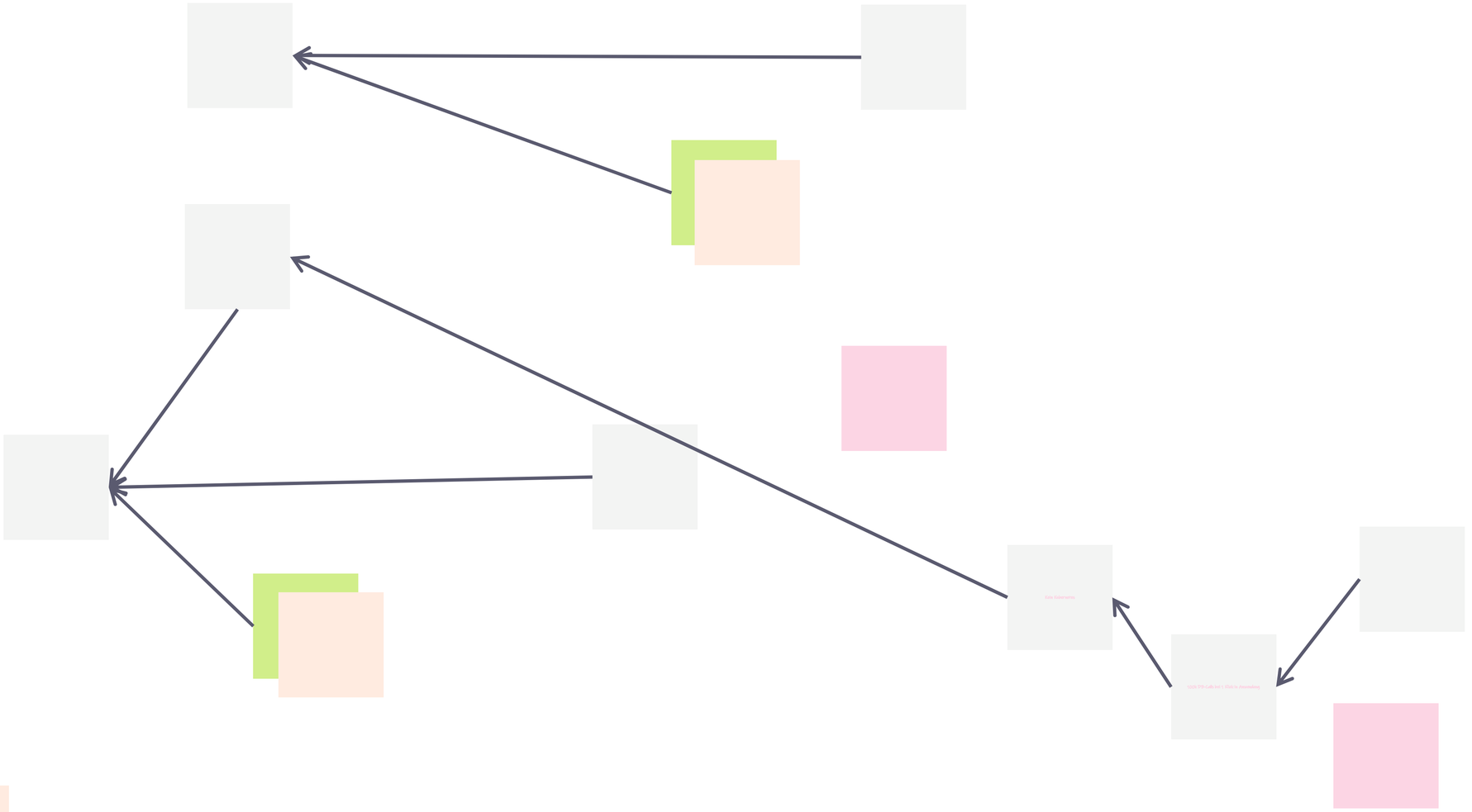
hängt ab von

Problem

Maß-
nahme

Keine
Maß-
nahme

Ergebnisse einer Problemsammlung



hängt ab von

Problem

Maß-
nahme

Gelöstes
Problem

Folge

Ergebnisse einer Problemsammlung

Bonuspunkte

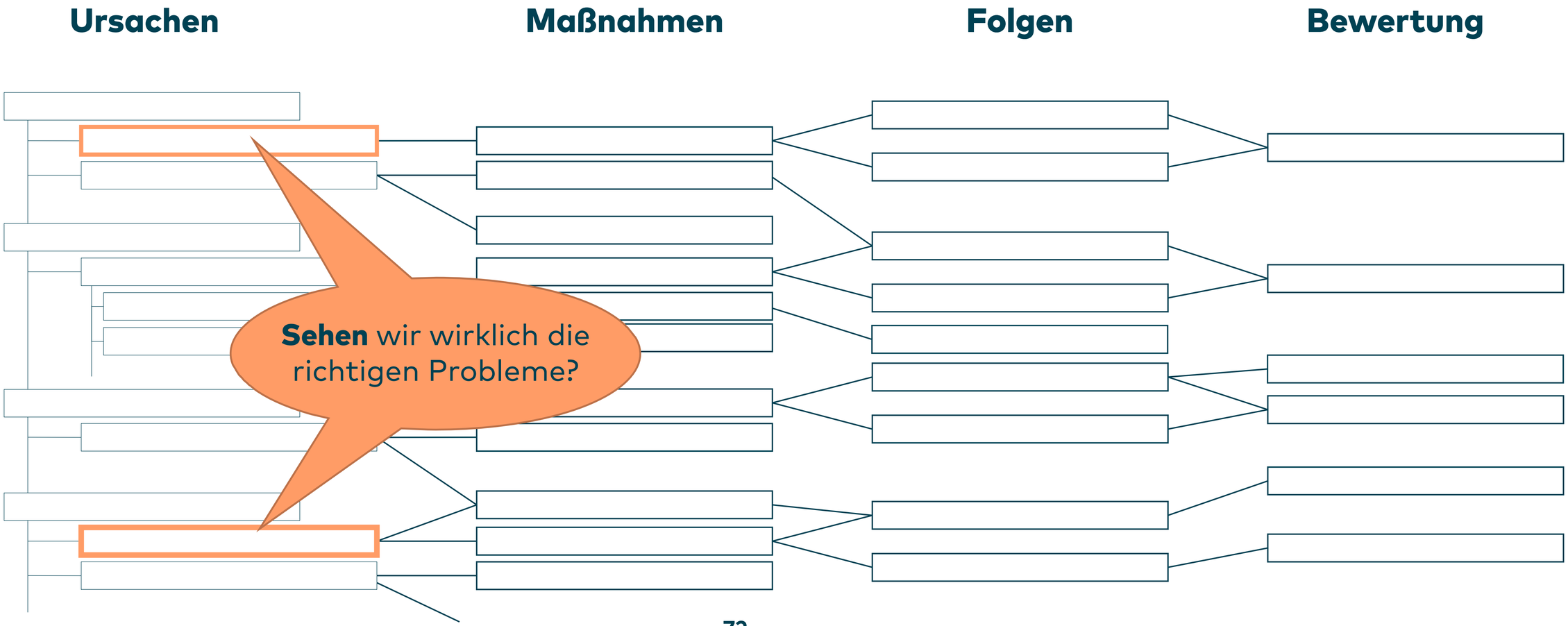
(nur kurz angerissen)

6. Punkt

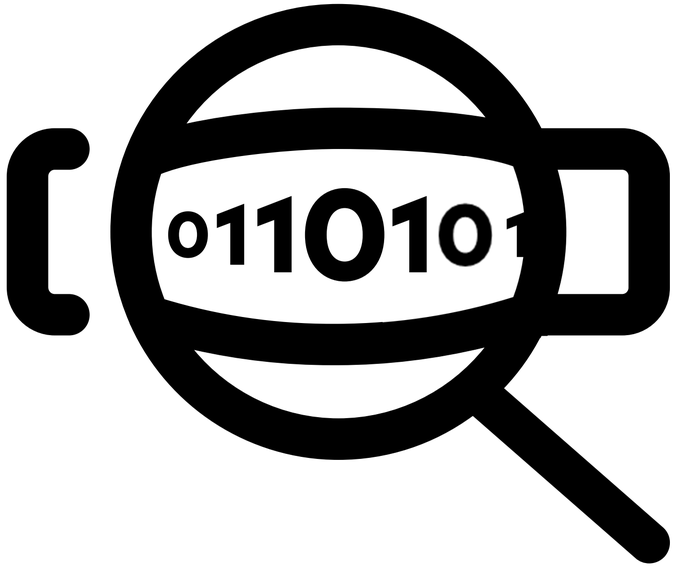
Denkfehler beachten

Sehen wir die wirklichen Probleme?

Problemidentifikation hinterfragen!



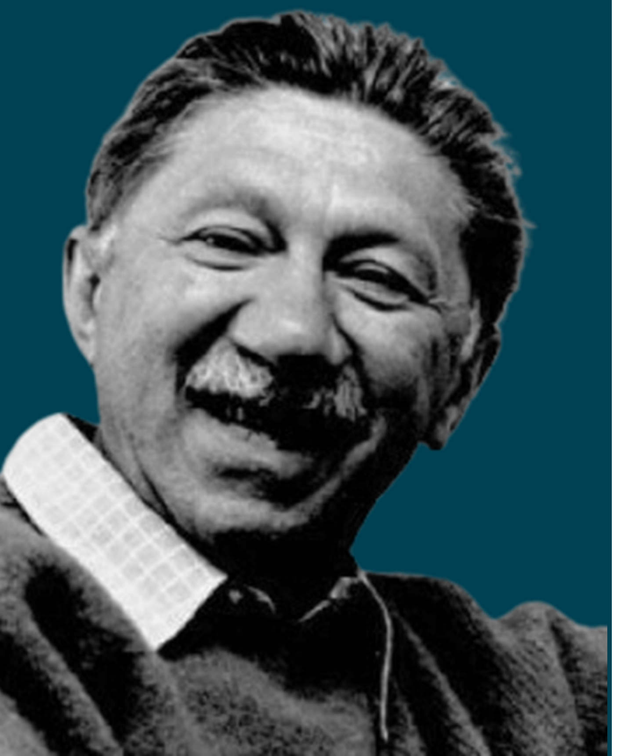
Wer nur im Code sucht,



wird auch nur dort Probleme finden!

Antimethode: Den Schlüssel unter der Straßenlaterne suchen

Nicht nur dort suchen,
wo das Licht hinfällt!

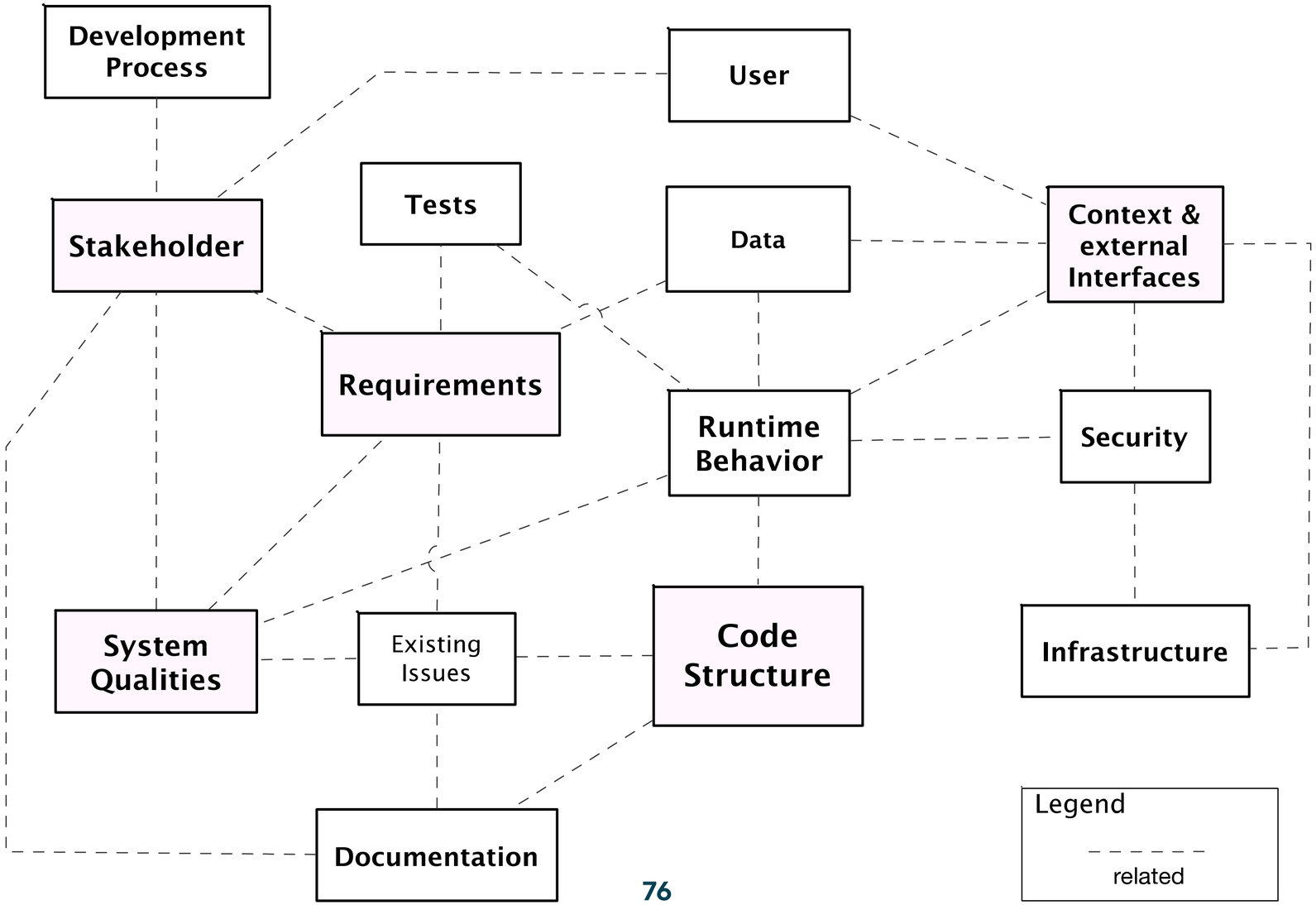


"I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."

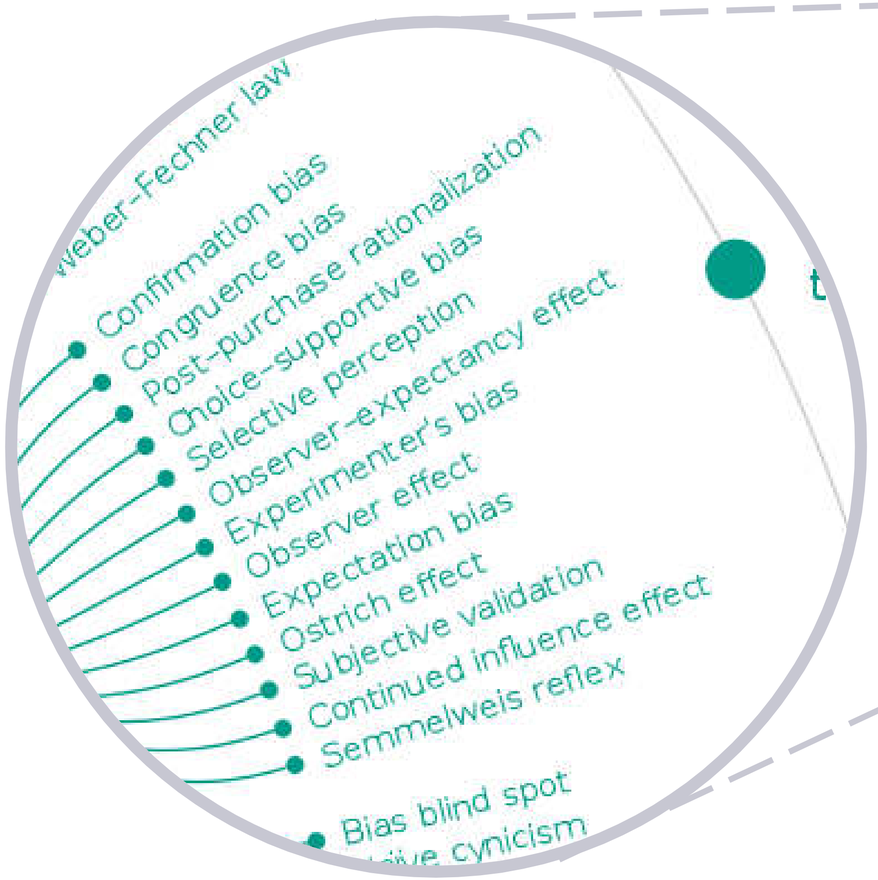
Abraham Maslow

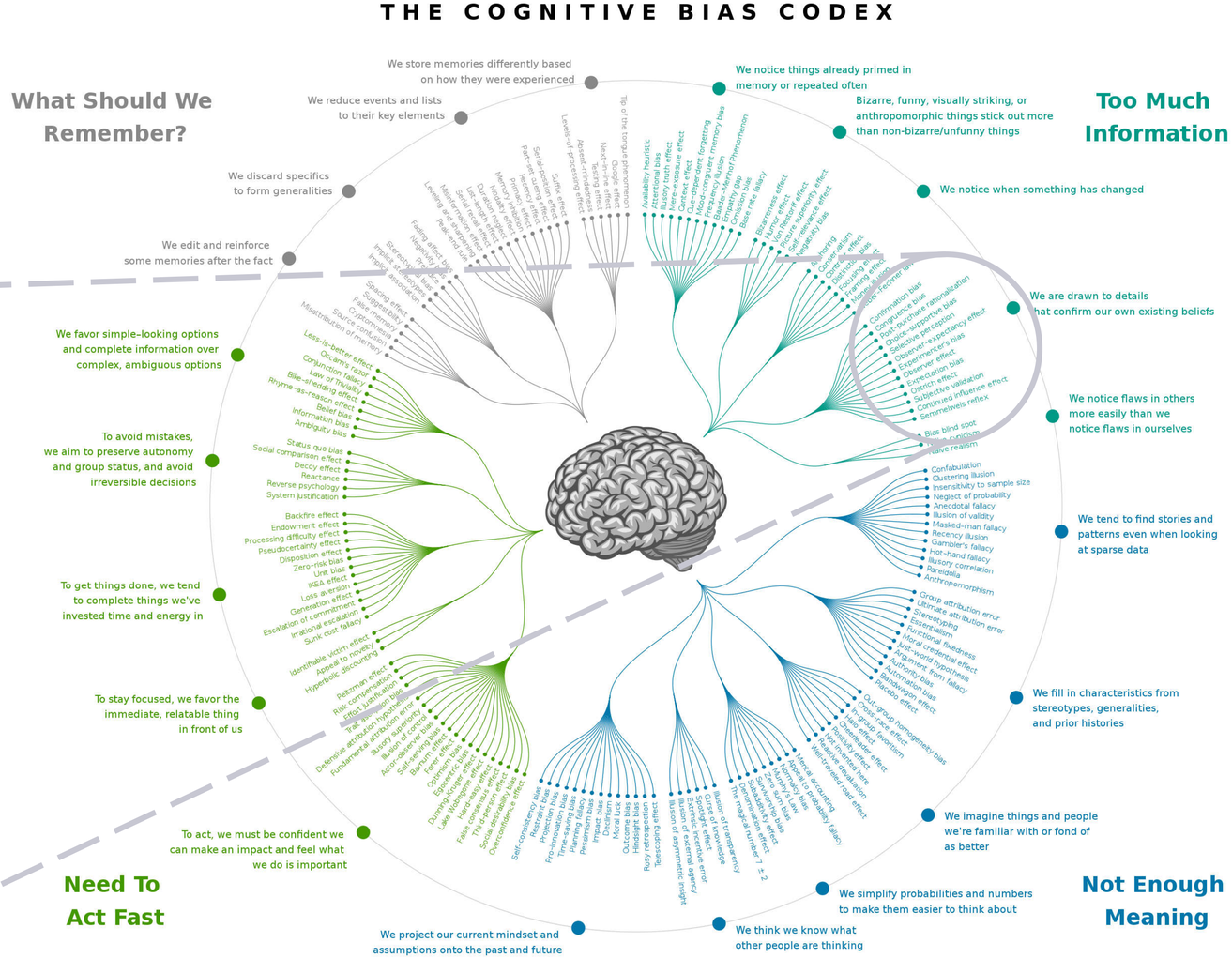
Bild: https://en.wikipedia.org/wiki/Abraham_Maslow#/media/File:Abraham_Maslow.jpg

Bewusst in die Breite gehen!



Die große Welt der Denkfehler





Ein paar Denkfehler als Beispiel

Recency Bias

Emotional Bias

Known Solution Bias

Grand Vision Bias

Sunk Cost Fallacy

Entnommen von Janelle Arty Starr: Idea Flow - How to Measure the PAIN in Software Development

Kausale Ketten & Netze in Frage stellen!

Wir haben keine anpassbare Login-Seite für unsere internen Anwendungen

Warum?

Wir haben kein Content Management System

...

...

Für welche Lösung ist das das Problem?

Lösungsraum

Problemraum

Warum (ist das ein Problem)?

Unsere IT-Mitarbeiter müssen ständig Anmeldeseiten für neue Anwendungen erstellen

Warum?

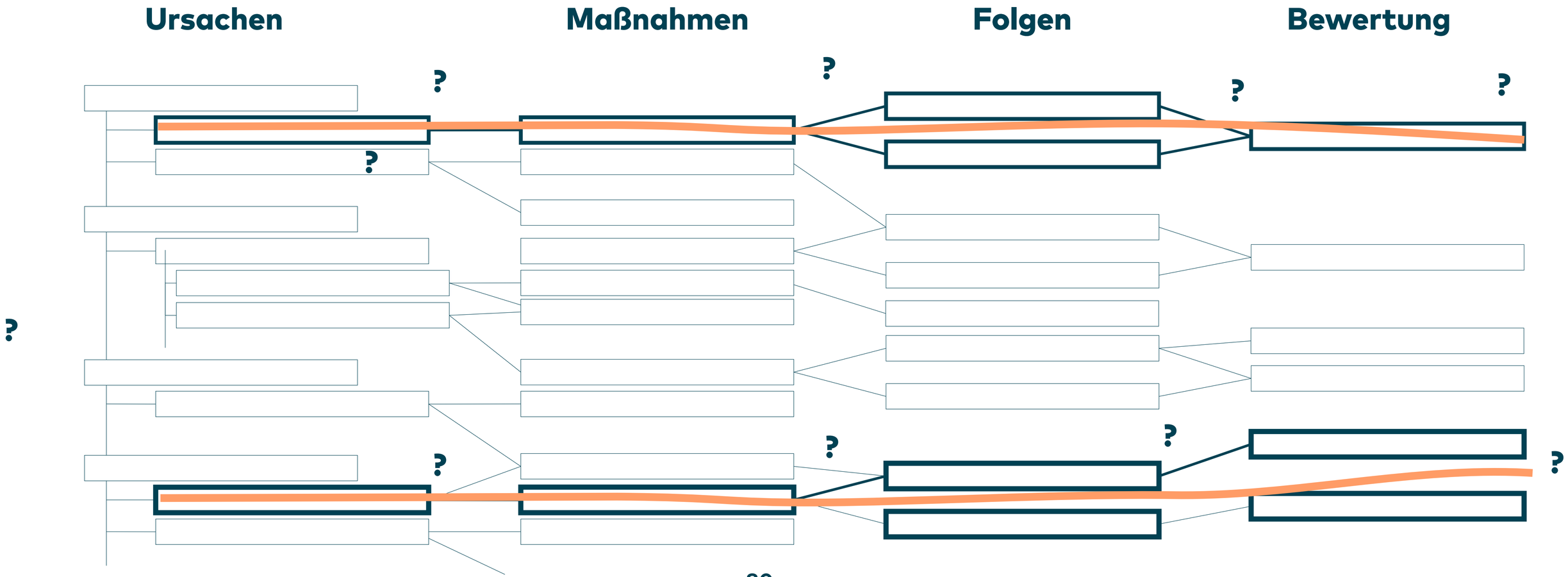
Zuständigkeiten und Budget-zuweisung für Themen der Rechteverwaltung sind unklar

Was noch?

Vorgeschriebenes Single Sign-On wurde bisher falsch bzw. nicht implementiert

Ist unser Weg der richtige?

Problemlösungsvorgehen reflektieren und Werkzeuge hinterfragen!



7. Punkt

Richtiges Problem?

„Problem… …oder Chance?“

Online-Lottoanbieter

Bei Rekordaus-schüttungen informieren wir unsere Kunden, dass sie bitte früher unser System nutzen sollen.

!

Miese Skalierbarkeit!

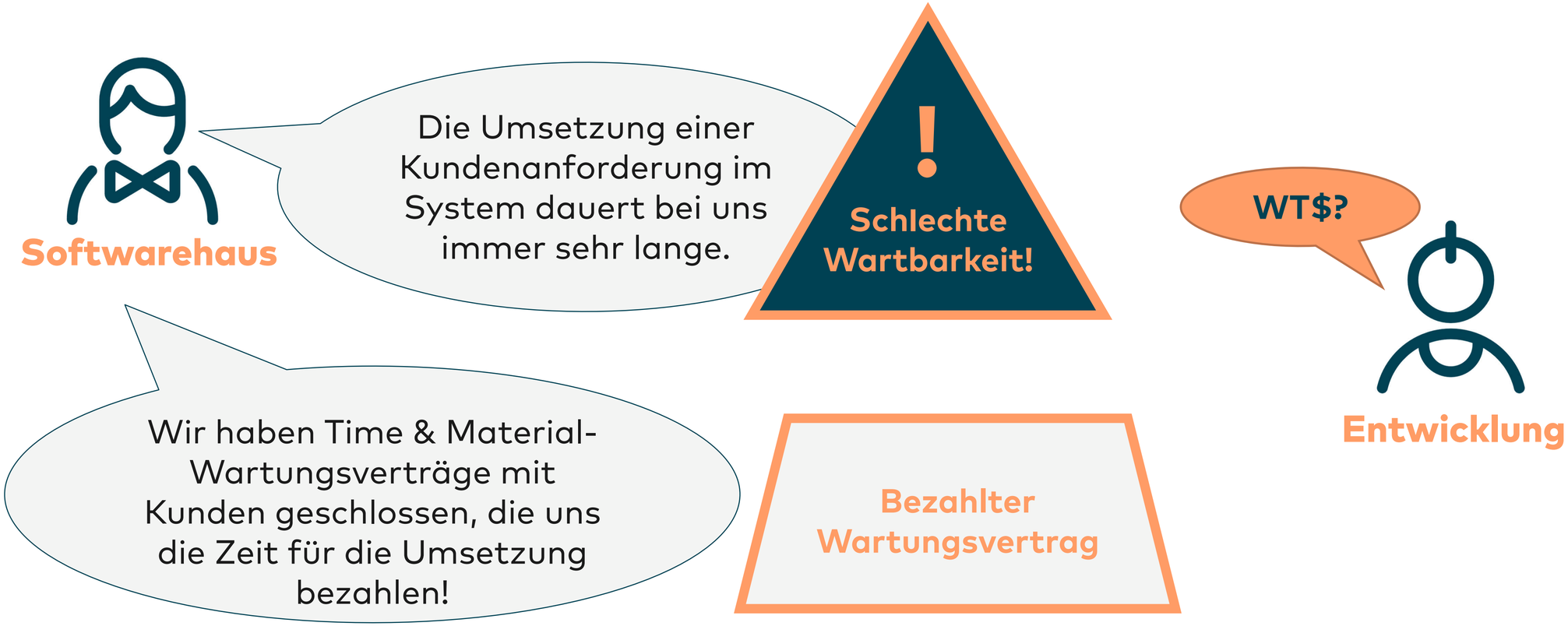
WT$?

Entwicklung

Vor großen Anstürmen können wir alle unsere Kunden anschreiben und damit noch einmal Werbung für unser Angebot machen!

Bessere Werbe-möglichkeiten!

„Problem… …oder Chance?“



„Problem… …oder Chance?“

Willkommenskultur von Problemen

Reaktionen

abgestritten

- **Anfeindungen** — *Probleme beweisen, keine Angriffspunkte liefern*
- **Widerstand** — *Support von einflussreichen Stakeholdern holen*
- **Verharmlosung** — *Konsequenzen der Probleme verständlich offenlegen*

ignoriert

- **Zweifel** — *Analysen nachvollziehbar kommunizieren*
- **Verblüffung** — *Klarheit über die untersuchten Probleme schaffen*
- **Gleichgültigkeit** — *Wichtige Eckdaten der Probleme kurz und bündig schildern*

akzeptiert

- **Zustimmung**
- **Begeisterung**

Tipps

https://aim42.github.io/#Expect-Denial

" Wenn Sie ein **Ziel** haben, und Sie sind auf dem Weg Ihr Ziel zu erreichen, dann haben Sie **kein Problem**!

Wenn Sie ein **Ziel** haben, und Sie sind **nicht** auf dem Weg Ihr Ziel zu erreichen, dann haben Sie **ein Problem**!

[..]

Ohne **Ziel** ist es nämlich verdam-mt schwer zu erkennen, ob man auf dem richtigen Weg ist."

Georg Jocham

Zitat aus https://georgjocham.com/apl006-wir-haben-ein-problem-haben-wir-wirklich-ein-problem/
Bild aus https://www.youtube.com/watch?v=q7JKlHJqDY4

Zusammenfassung

Wie löse ich knifflige Probleme in Softwaresystemen?

Überblick der Punkte

Reihenfolge ist irrelevant!

Wiederholungen immer und immer wieder notwendig!

1. Was sind die Probleme?
2. Problem(rück|vor)verschiebung
3. Problemverständnis
4. ProblemursachEN
5. Passende Maßnahmen
6. Denkfehler
7. Richtiges Problem?

Stark inspiriert von Walter Schönwandts Buch „Knifflige Probleme lösen“

Die Lösungen für knifflige Probleme

… und warum es keine eindeutige Antwort drauf gibt

„Nur Herr Meier (extern) kennt sich noch mit unserer Software aus!“

Keiner steigt im Quellcode durch

Wir haben den Sourcecode nicht

Niemand hat Lust auf C

Zu wenig Entwickler

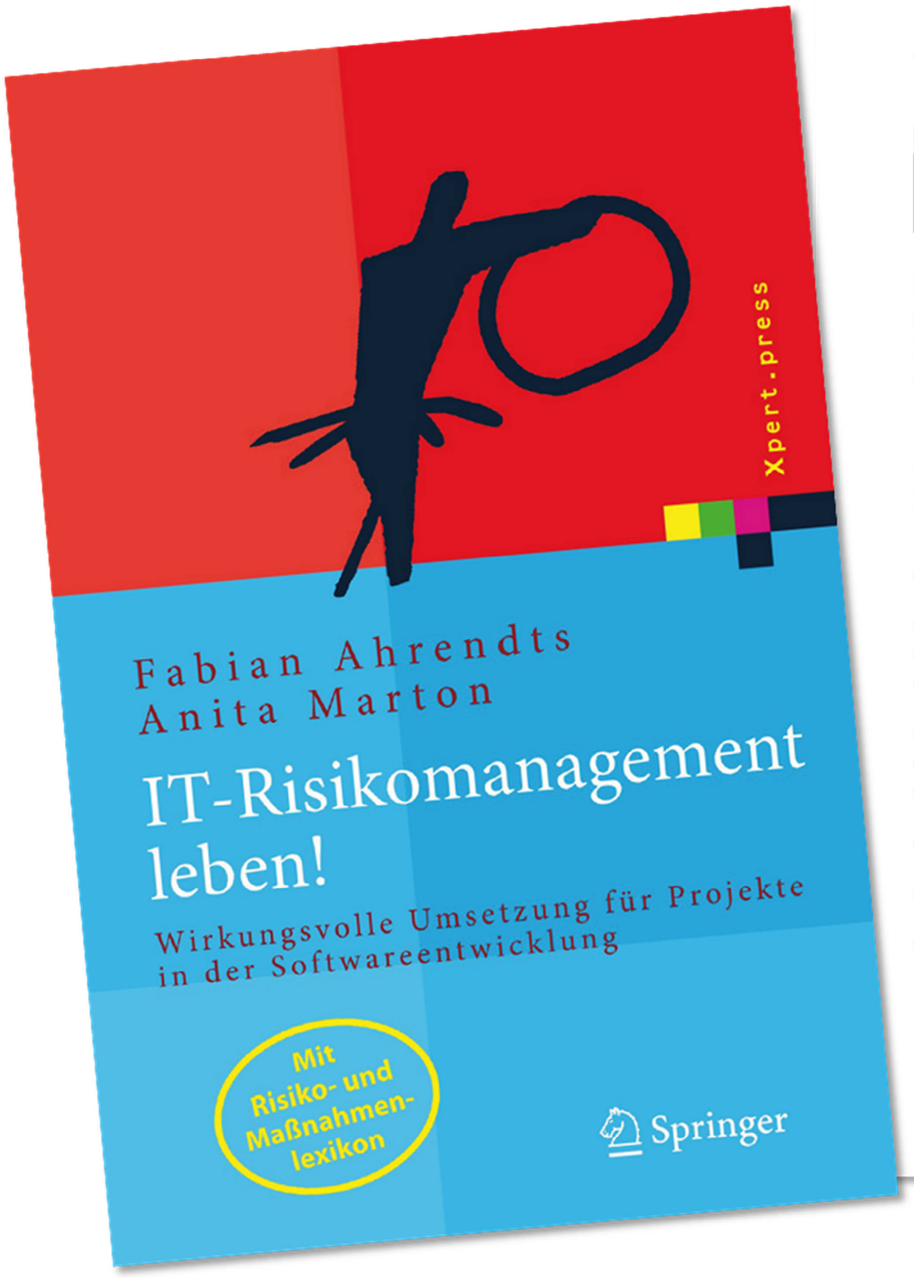
Herr Meier teilt sein Wissen nicht

Keine Zeit fürs Dokumen-tieren

Software wird in 3 Monaten abgelöst

Ein paar Lesetipps

für die individuelle Lösungen



7.10.7 Unmotivierte Mitarbeiter

KATEGORIE: MENSCH

P A D I T A

Phasenübergreifend

☑ Kleinprojekte
☐ Projektvorbetrachtung

Risikosatz

Aus ungenügender Erfahrung des Projektleiters, keine eindeutige Zuordnung von Aufgaben, Rollen und Verantwortlichkeiten, unzureichende Einarbeitung neuer Mitarbeiter, ungenügende Kommunikation innerhalb des Projektteams und zwischen dem Projektleiter und den Projektmitarbeitern könnte die **Motivation der Mitarbeiter sinken**, was zu einer Erhöhung der Mitarbeiterfluktuation, zu einem Kommunikationsrisiko und einer mangelhaften Arbeitsleistung/Produktivität führen kann.

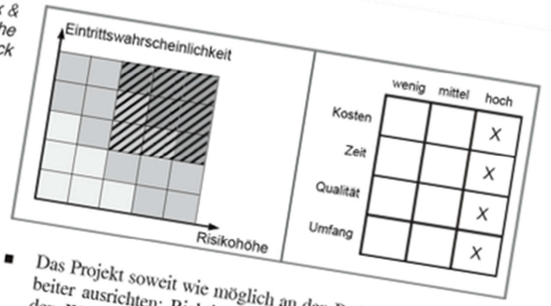
Risikobeschreibung

Die Motivation des Projektteams ist einer der Erfolgsfaktoren für die Produktivität eines Projektes und die Qualität der zu erstellenden Software. Maßnahmen zu Erhöhung der Motivation werden aber nur selten in der Projektplanung berücksichtigt, insbesondere weil Motivation nur schwer messbar ist.

Indikatoren

- Das Projektteam hält sich nicht an das vereinbarte Vorgehensmodell
- Frustration bei den Projektbeteiligten.
- Unangebrachter Zynismus und Sarkasmus auf dem Projekt.
- Arbeitspakete werden nicht in der möglichen Geschwindigkeit und der notwendigen Qualität erledigt.
- Häufige Krankheitsmeldungen.
- Mobbing zwischen einzelnen Projektmitarbeitern.
- Wenig Kommunikation innerhalb des Projektteams und zum Projektleiter.
- Keine eindeutige Aufgaben- und Zieldefinitionen für die Mitarbeiter. Die Mitarbeiter kenne ihre Rolle im Projekt nicht.
- Mitarbeiter können ihren Einfluss auf den Projekterfolg nicht benennen.
- Menschliche Bedürfnisse werden durch das Projekt und das Projektmanagement nicht berücksichtigt.

Risikomatrix & Das magische Viereck



Maßnahmen

- Das Projekt soweit wie möglich an den Bedürfnissen der Mitarbeiter ausrichten: Richtige Stelle für die richtige Person, nach den Wünschen/Erwartungen der Mitarbeiter fragen und diese berücksichtigen.
- Mitarbeiter in ihrer Arbeit und den Aufgaben unterstützten: Produktive Arbeitsplätze, geeignete Werkzeuge, motivationsfördernde Maßnahmen (persönliches Wachstum, Fortbildung und Schulungen) einleiten und dadurch Kompetenzen des Einzelnen entwickeln.
- Teambildung fördern: Gemeinsame Unternehmungen und Freizeitaktivitäten anbieten.
- Klaren Ergebnisabsprachen treffen, gegenseitige Erwartungen klar formulieren.
- Kommunikations- und Eskalationswege bereitstellen (siehe auch Maßnahmen „Kommunikationsrisiko innerhalb des Projektes").
- Kritikgespräche nach der Sandwichtechnik führen: Positiver Beginn. Äußerung der Kritik auf Basis von Fakten und Sachverhalten, positives Ende durch Sammlung von Lösungsmöglichkeiten und gemeinsames Festlegen von Konsequenzen für die Zukunft.
- **Motivation durch Kommunikation**: Eine gute und offene Kommunikation innerhalb des Projektteams und zum Projektleiter schafft eine vertrauensvolle Atmosphäre und baut eine Beziehung zwischen allen Projektbeteiligten auf. Der Projektleiter kann dadurch immer auf den aktuellen Stand bleiben und weiß, was im Team gerade an aktuellen Themen ansteht.
- **Motivation durch Ziele**: Den Projektmitarbeitern die Aufgaben und Ziele für das aktuelle Projekt mitteilen, ggf. können Projektprämien vereinbart werden. Ziele so wählen, dass sie realis-

CLOUD NATIVE TRANSFORMATION PATTERNS

Tools for creating effective Cloud Native architecture—and remaking the way we work

WHAT ARE CLOUD NATIVE PATTERNS?

When it comes to Cloud Native, most of us are junior. The technology is so new, and our understanding of the architecture is constantly evolving. Sharing our rapidly growing knowledge is essential. Creating Cloud Native-specific patterns is a way to name the things we are learning as developers, engineers, and technology managers so we can talk to each other more effectively. As we learn and continue, the patterns will evolve and improve alongside our understanding.

Patterns are not a hack. They are not a quick and easy way to solve difficult problems without careful thought. Instead, they are a language for sharing context-specific working solutions. And here is the place to find and explore Cloud Native patterns–and, soon, share some of your own. **This site is intended to be the home of a Cloud Native patterns community, the place to explore existing patterns and collaborate on creating new ones.** We openly welcome questions, conversation, and contributions!

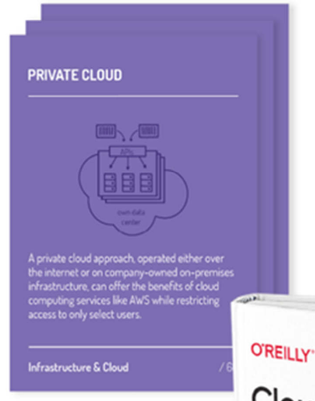
Explore patterns

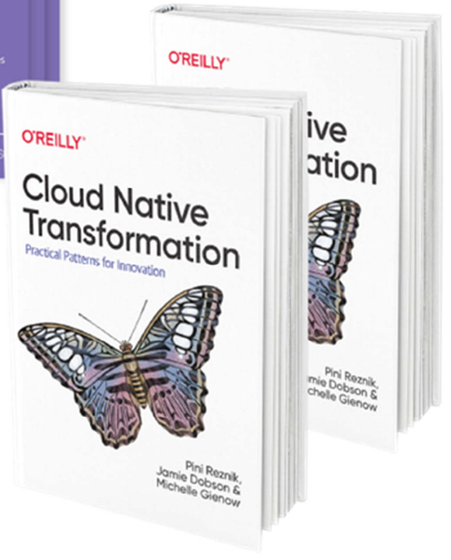
Strategy & Risk Reduction

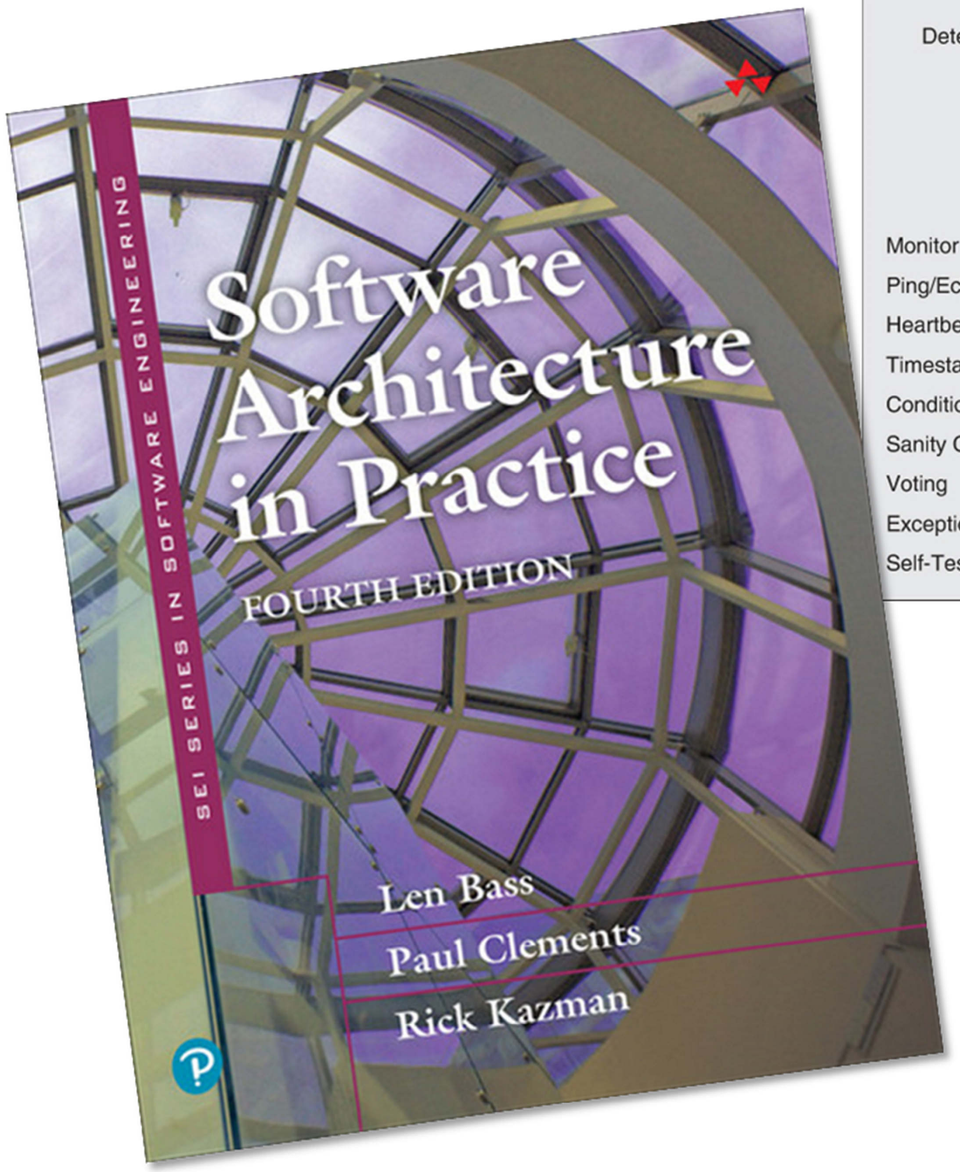
Organization & Culture

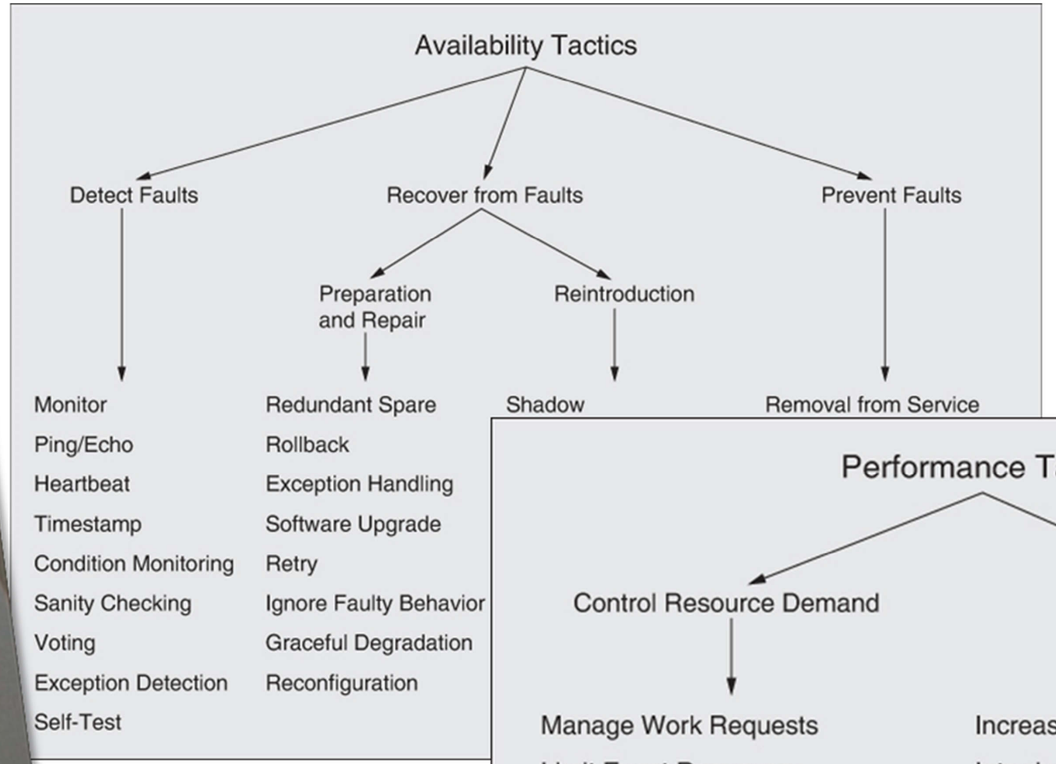
Development & Design

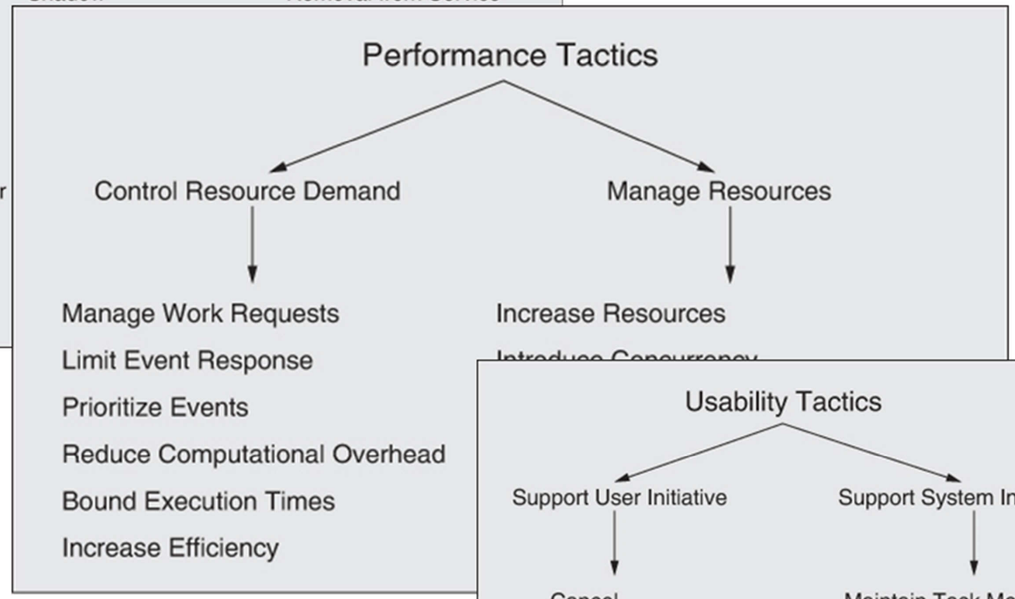
Infrastructure & Cloud

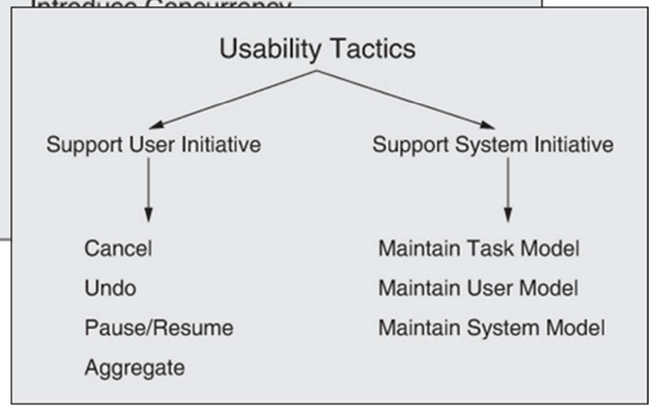


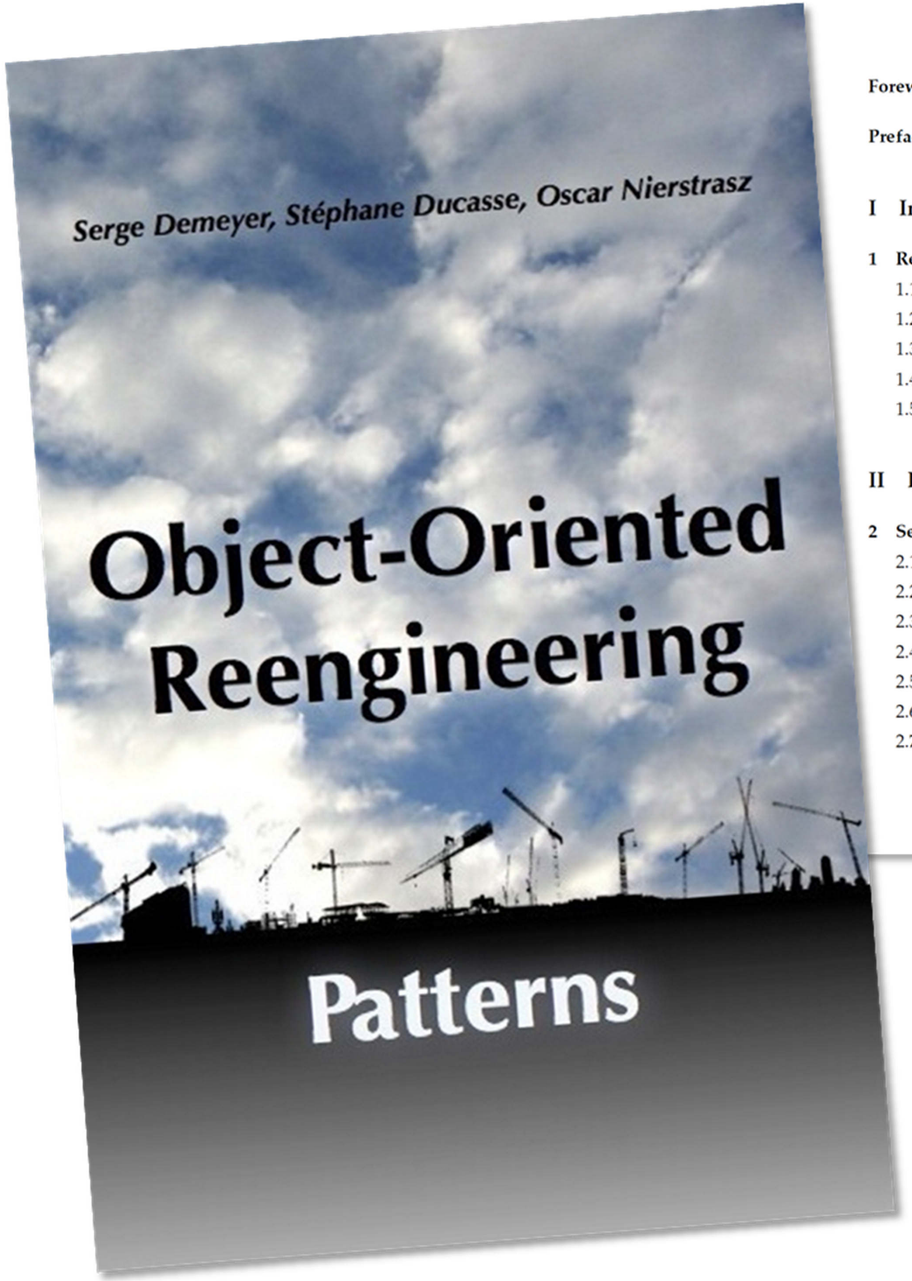



SEI SERIES IN SOFTWARE ENGINEERING
Software Architecture in Practice
FOURTH EDITION
Len Bass
Paul Clements
Rick Kazman


Availability Tactics
Detect Faults
Recover from Faults
Prevent Faults
Preparation and Repair
Reintroduction
Monitor
Ping/Echo
Heartbeat
Timestamp
Condition Monitoring
Sanity Checking
Voting
Exception Detection
Self-Test
Redundant Spare
Rollback
Exception Handling
Software Upgrade
Retry
Ignore Faulty Behavior
Graceful Degradation
Reconfiguration
Shadow
Removal from Service


Performance Tactics
Control Resource Demand
Manage Resources
Manage Work Requests
Limit Event Response
Prioritize Events
Reduce Computational Overhead
Bound Execution Times
Increase Efficiency
Increase Resources


Usability Tactics
Support User Initiative
Support System Initiative
Cancel
Undo
Pause/Resume
Aggregate
Maintain Task Model
Maintain User Model
Maintain System Model



Contents

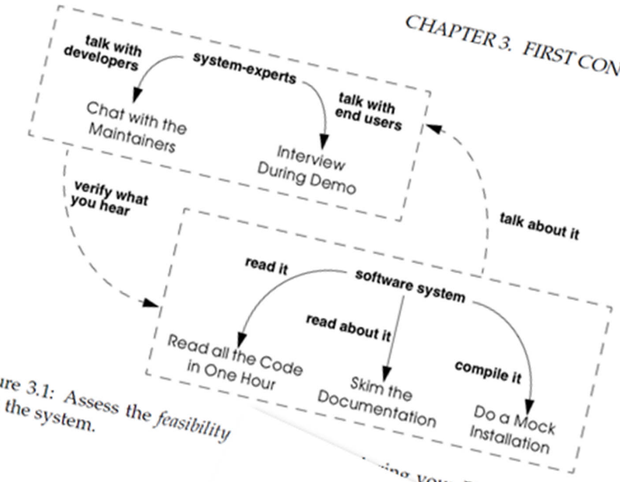


Figure 3.1: Assess the *feasibility* with the system.

based on that knowledge plar
cancel the project.

The patterns Chat with t
will help you get acquainte
thumb, spend four days t
the week to compile all th
is no strict order in which
gested by the sequence ir
have often find ourselves
of the necessity to doub
with the maintainers we
ask questions about wh
Hour and Skim the Dc
check the source code

In certain situatio
applicable due to a l
have left the compa
tain systems lack ar
an Interview Durin
lem, because some
anyway. Howeve
and it should be r

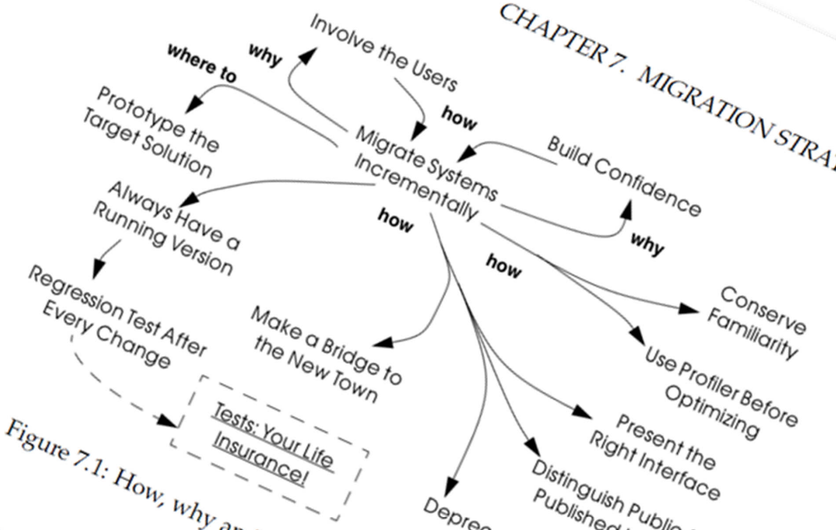
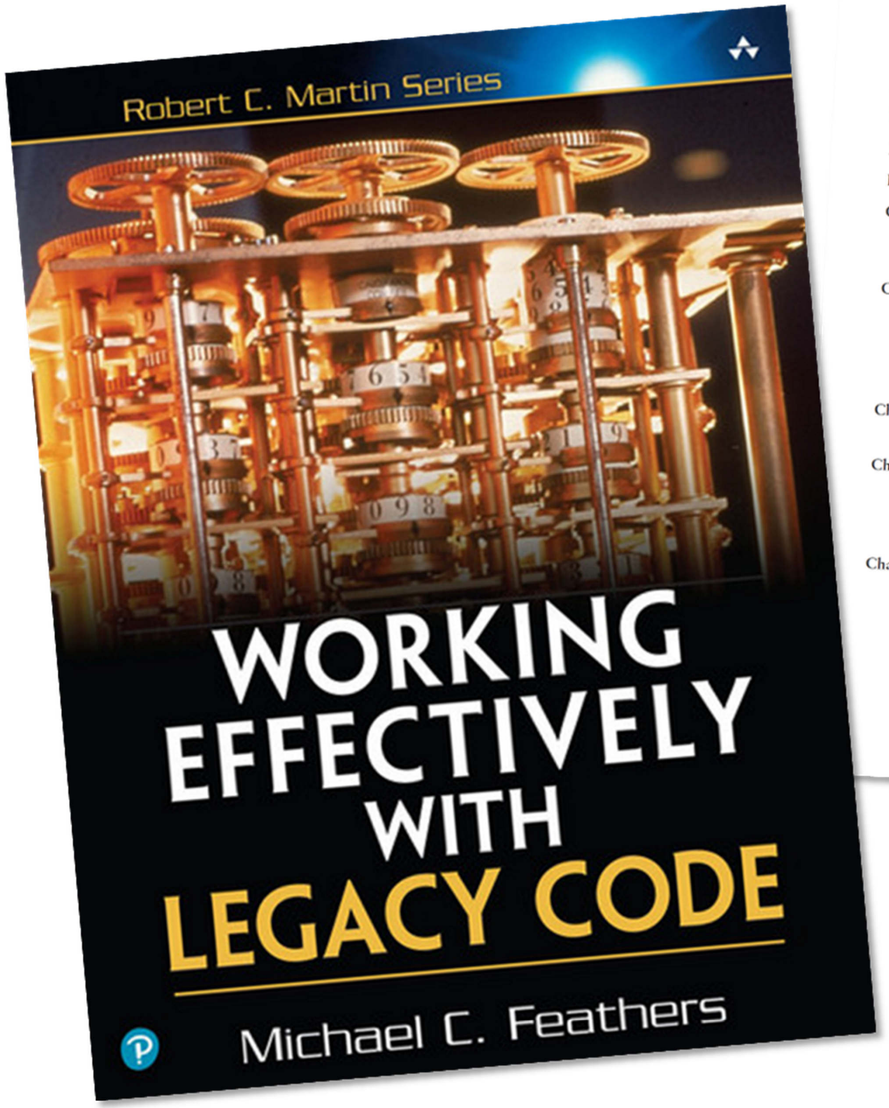


Figure 7.1: How, why and whither to migrate legacy sys

of the users, and you must adopt a strat
painlessly from the existing system
system.

The central messa
This is, howeve
der to Mi
of



Contents vii

Contents

viii Contents

Contents ix

x Contents

Mehr Infos zum großen Thema

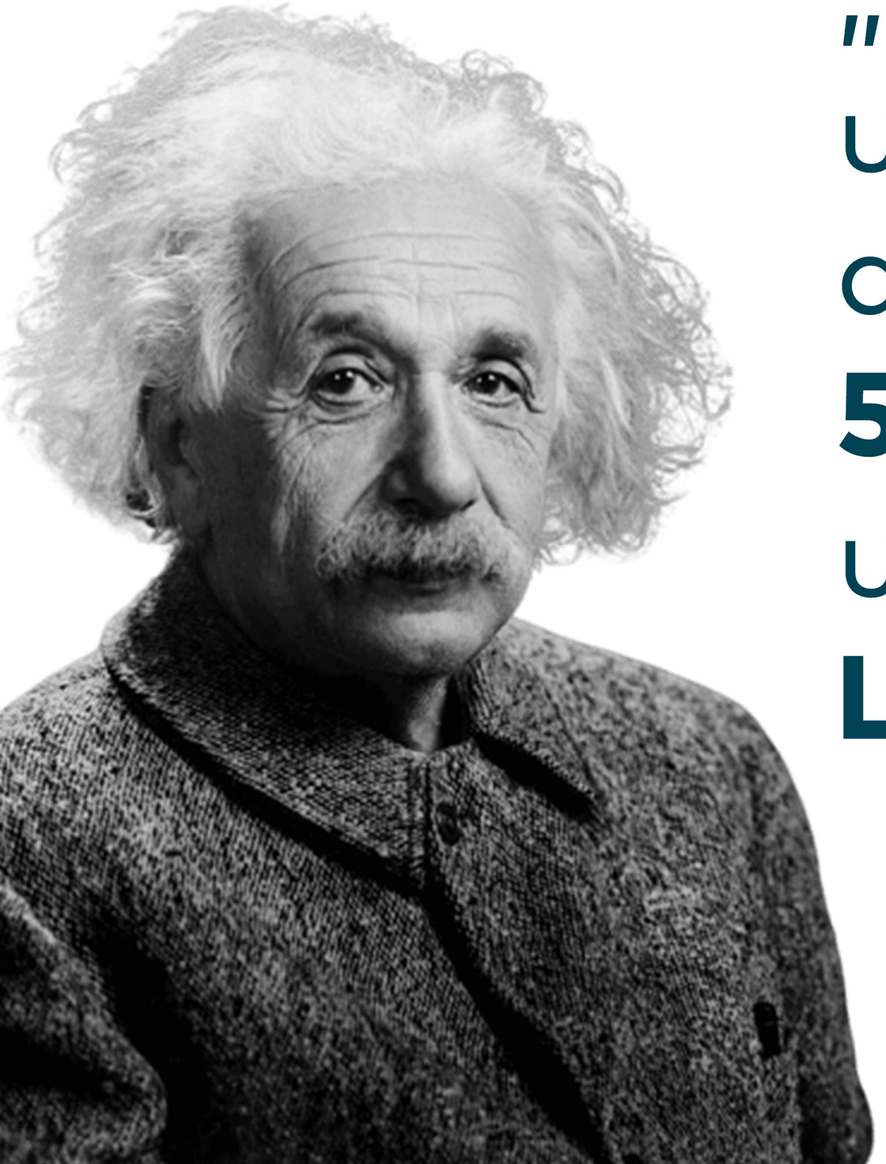
Kuratierte Liste mit Tipps zum Umgang mit Legacy Systeme

Awesome Legacy Systems awesome

> A curated list of awesome resources and links about tackling legacy systems that gives hope.

Hi! My name is Markus Harrer. I created this repository to share a set of resources and links that I found valuable and inspiring while working with legacy systems. I hope that you'll like it, too! If you want to add something, feel free to contribute your ideas. You can also support the community in several ways!

https://github.com/feststelltaste/awesome-legacy-systems



„Wenn ich **eine Stunde** habe, um ein Problem zu lösen, dann beschäftige ich mich **55 Minuten** mit dem **Problem** und **5 Minuten** mit der **Lösung**."

Albert Einstein

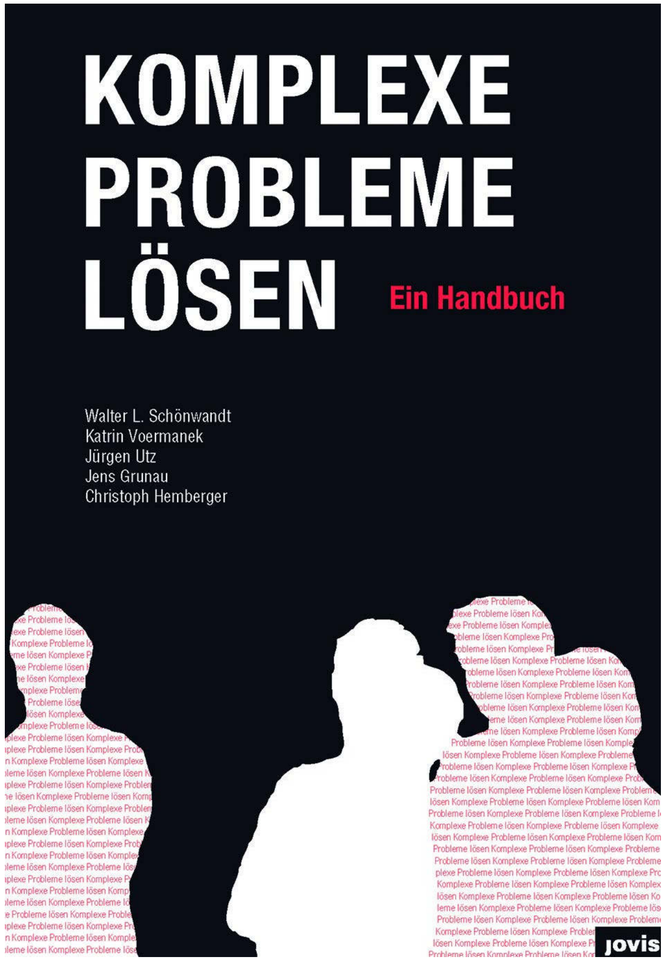
https://commons.wikimedia.org/wiki/File:Albert_Einstein_Head_cleaned.jpg

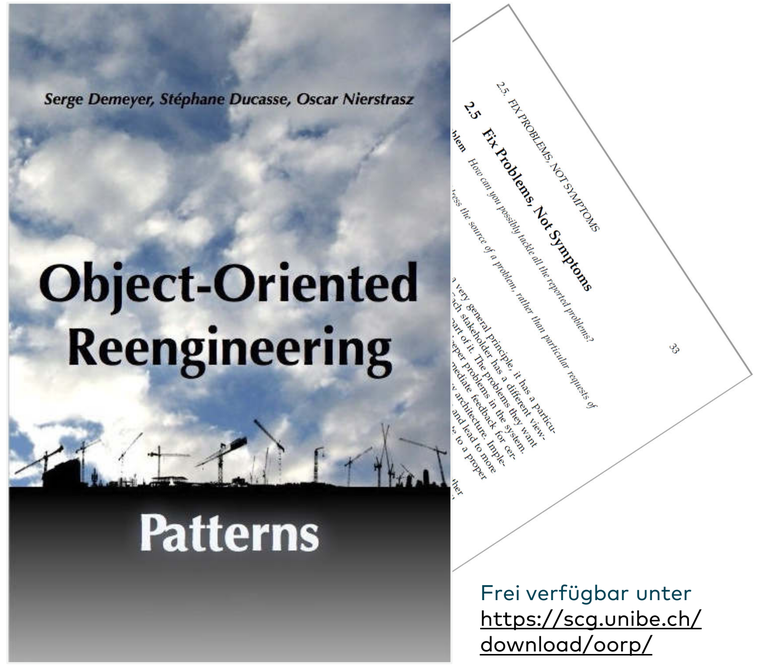
Kudos

Dieser Talk baut vor allem auf die Ideen von Prof. Walter Schönwandt et.al. auf*

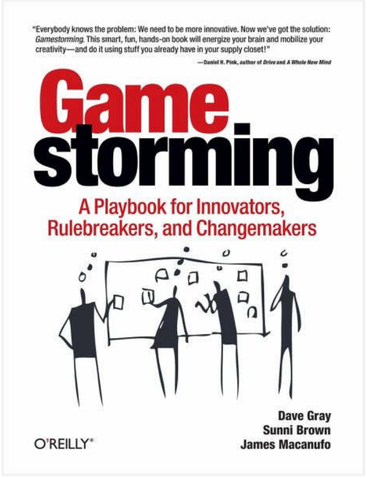
Podcast-Folge von Georg Jocham
Abenteuer Probleme lösen, Folge 021: So löst man komplexe Probleme, im Gespräch mit Prof. Walter Schönwandt
https://georgjocham.com/apl-020-so-loest-man-komplexe-probleme-im-gespraech-mit-prof-dr-walter-schoenwandt/

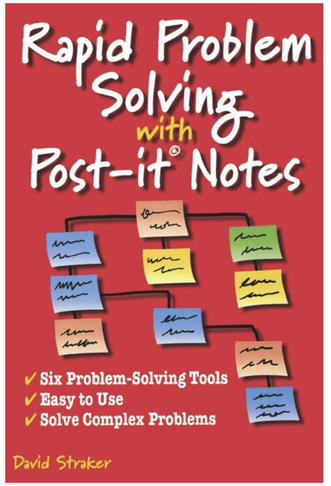
game changer





Frei verfügbar unter https://scg.unibe.ch/download/oorp/

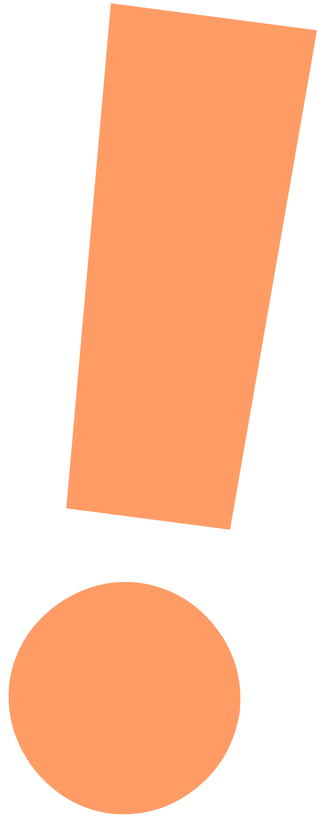




usual suspects

* es gibt aber genügend Praxiserfahrungen damit!

Vielen Dank



Fragen

+ Antworten

INNOQ

UNSER ANGEBOT

Produktkonzeption & Design
Software-Entwicklung & -Architektur
Technologie-Beratung
Infrastruktur & Betrieb
Wissenstransfer, Coaching & Trainings

FAKTEN

~160 Mitarbeitende
1998 gegründet
9 Standorte in
D & CH

FOKUS

Webapplikationen
SaaS
IoT
Produktentwicklung
ML/AI
Blockchain

TECHNOLOGIEN

(Auswahl)

Java/Spring
Ruby/Rails
Scala
AWS
Kubernetes
Azure
JavaScript
Python
C#
ML/AI
Blockchain

KLIENTEN

Finance • Telko • Logistik • E-Commerce • Fortune 500 • KMUs • Startups

AXA breuninger ebay 101 mobile.de SACAC VORWERK