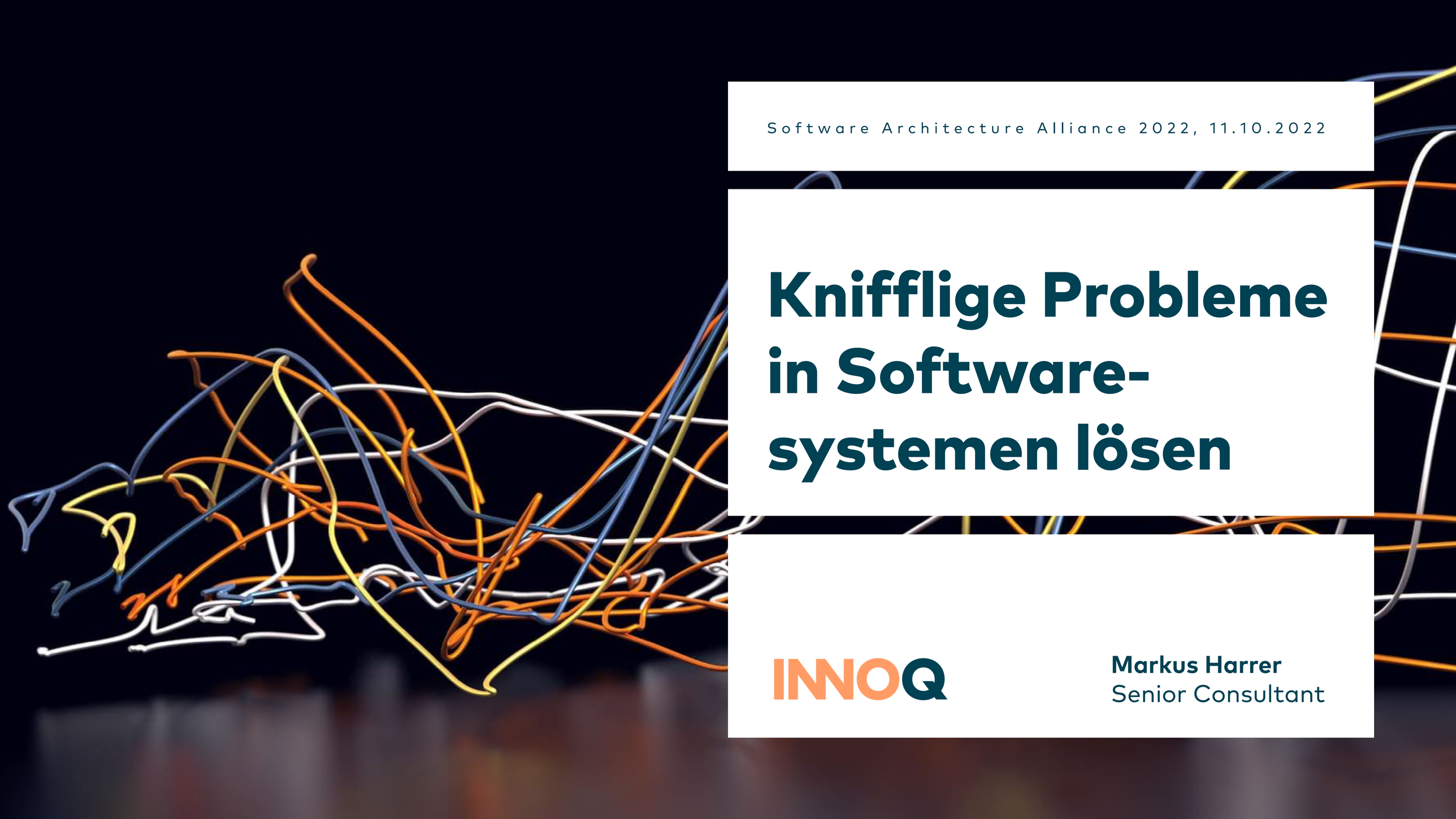


Software Architecture Alliance 2022, 11.10.2022

Knifflige Probleme in Software-systemen lösen

INNOQ

Markus Harrer
Senior Consultant

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

Markus Harrer

Senior Consultant / Nürnberg, Deutschland

- Softwarearchitektur-Entwicklung und -Bewertung
- Software-Modernisierung und -Rightsizing
- Datenanalysen in der Softwareentwicklung

aim42

cards42.org

Cards for Analyzing and Reflecting on Doomed Software

iSAQB ACCREDITED Trainer

Foundation & IMPROVE

inkl. Lehrplan

DataCamp Instructor

https://feststelltaste.de/

https://softwareanalytics.de

Ziele und Scope des Vortrags

„Knifflige Probleme in Softwaresystemen lösen“

Ziel 1: Lösungsreflexe unterbinden

Schnelles Denken

Symptom

~95%

„Abhilfe“

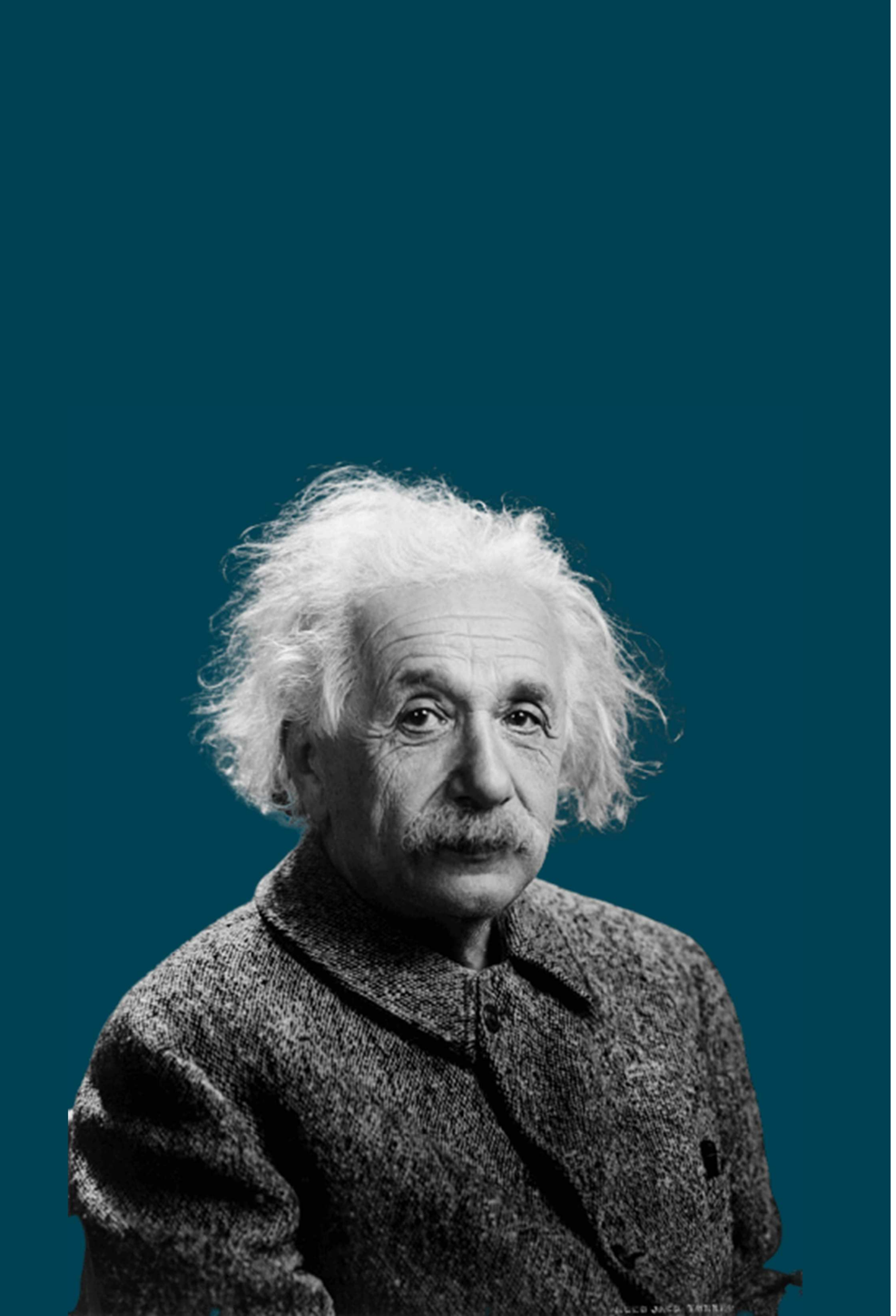
L a n g s a m e s Denken

Problem

Lösung

Ziel 2: Probleme nachvollziehbar lösen

Euch helfen, Fäden zwischen Problemen und Lösungen zu ziehen



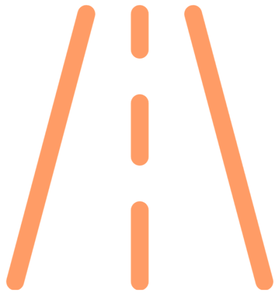
"Das Problem zu erkennen ist wichtiger als die Lösung zu erkennen, denn die genaue Darstellung des Problems führt zur Lösung."

Albert Einstein

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

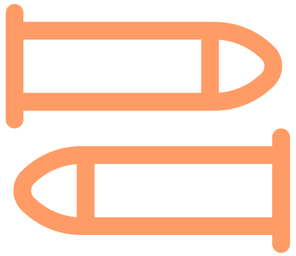
Nicht-Inhalte des Talks

… aber zumindest bietet dieser Talk Input dafür



Roadmap-Erstellung

Portfoliomanagment



Silver Bullets

Knifflige Probleme

in Softwaresystemen

Kniffliges Problem

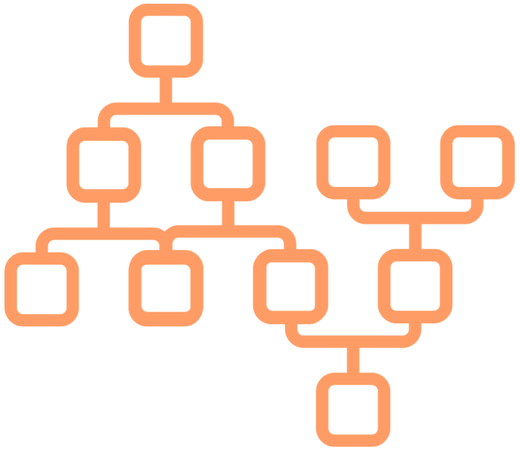
Richtiges Schlammassel

Vielfältig
Intransparent
Viele Abhängigkeiten
Wechselwirkungen

Beispiele für knifflige Probleme

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

„Die Suche nach Verträgen dauert 5 Minuten!“



„Angebot berechnen fabriziert 12000 Service-Calls!“

„Nur Herbert kennt sich mit der Abrechnung aus!“

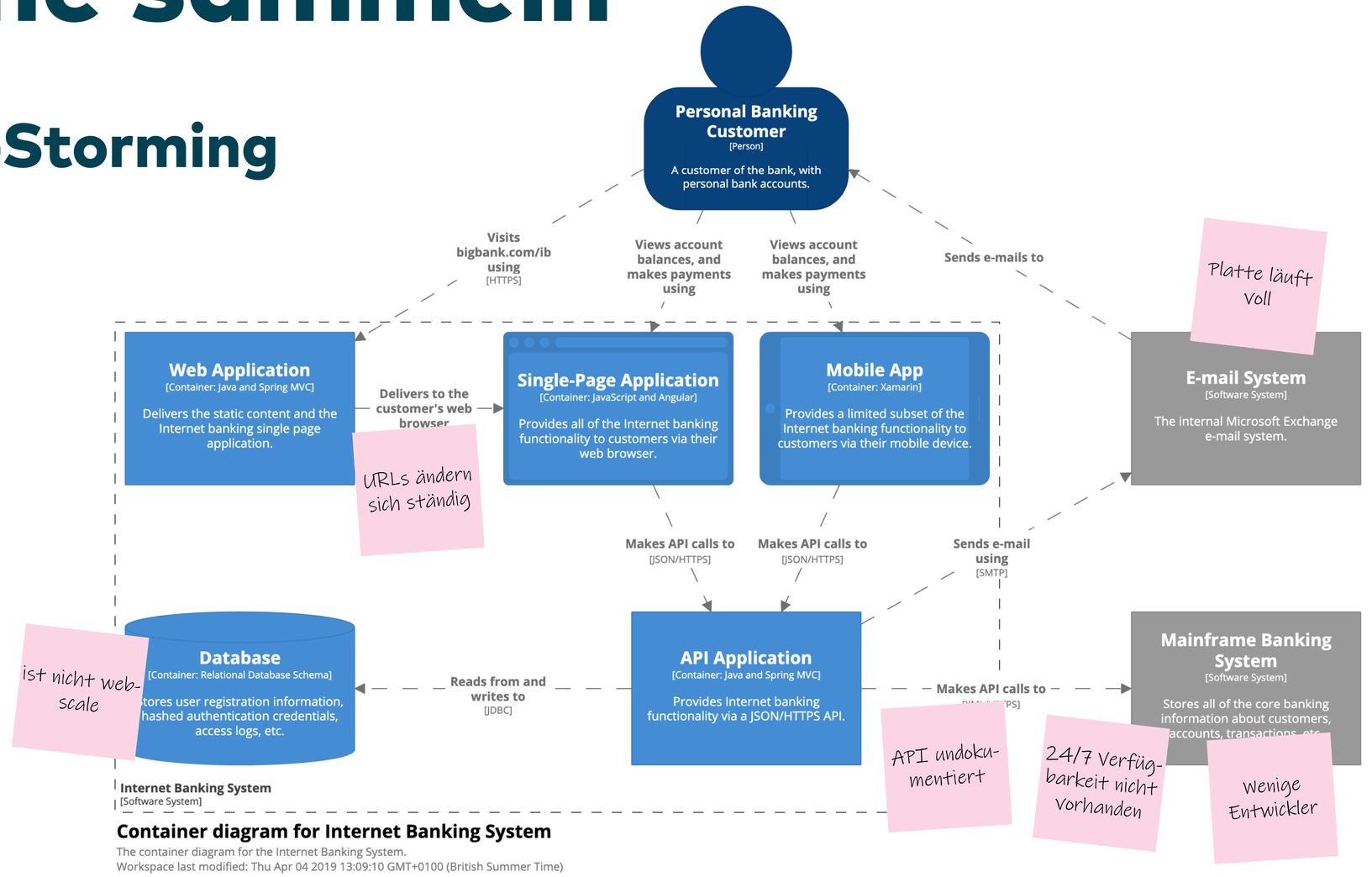
Probleme sammeln

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

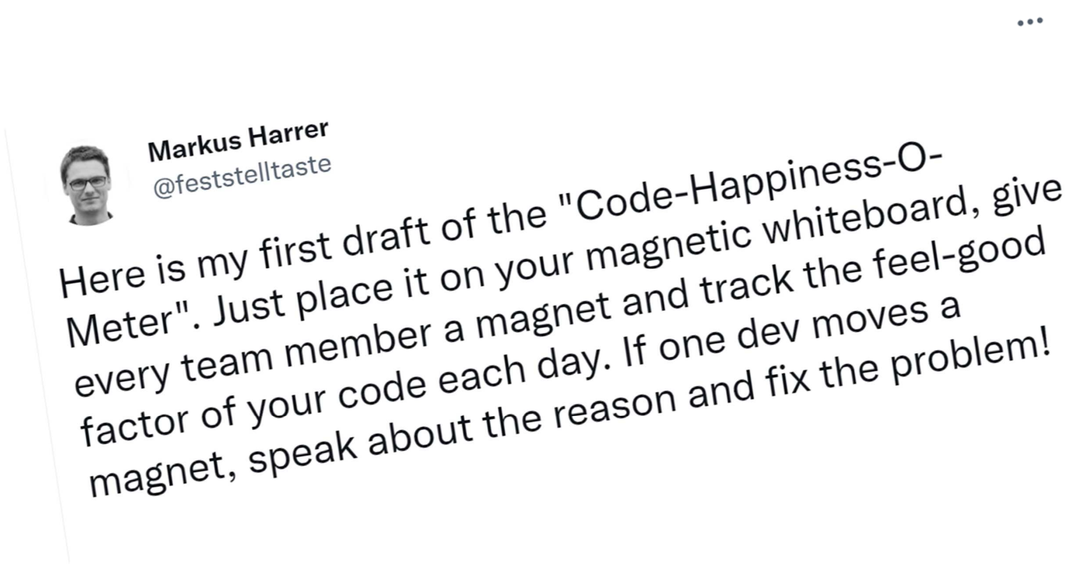
Beispiel: Risk-Storming



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

Probleme sammeln

Beispiel: Code-Happiness-O-Meter



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

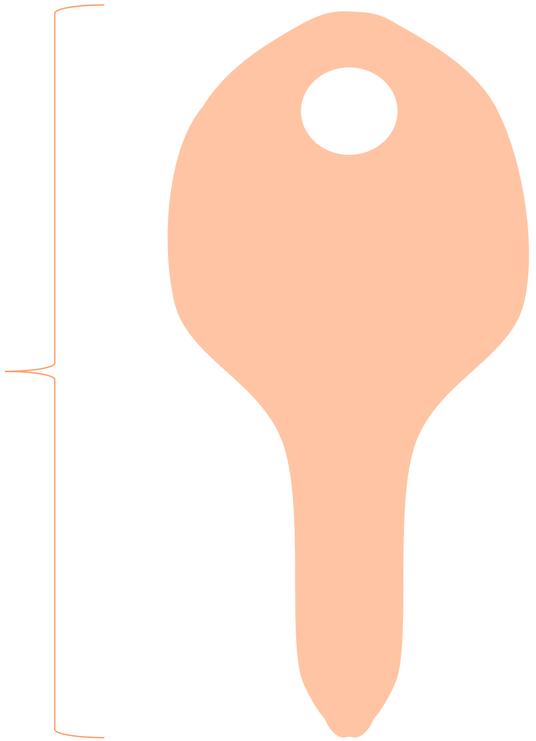
Ergebnisse einer Problemsammlung

Problem

Was machen wir nun damit?

Problem lösen (frei nach Prof. Walter Schönwandts „Key Seven“)

Ungefähres Verhältnis der Zeitdauer der Themen



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

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

Gelöstes Problem

Ergebnisse einer Problemsammlung

Problem

2. Punkt

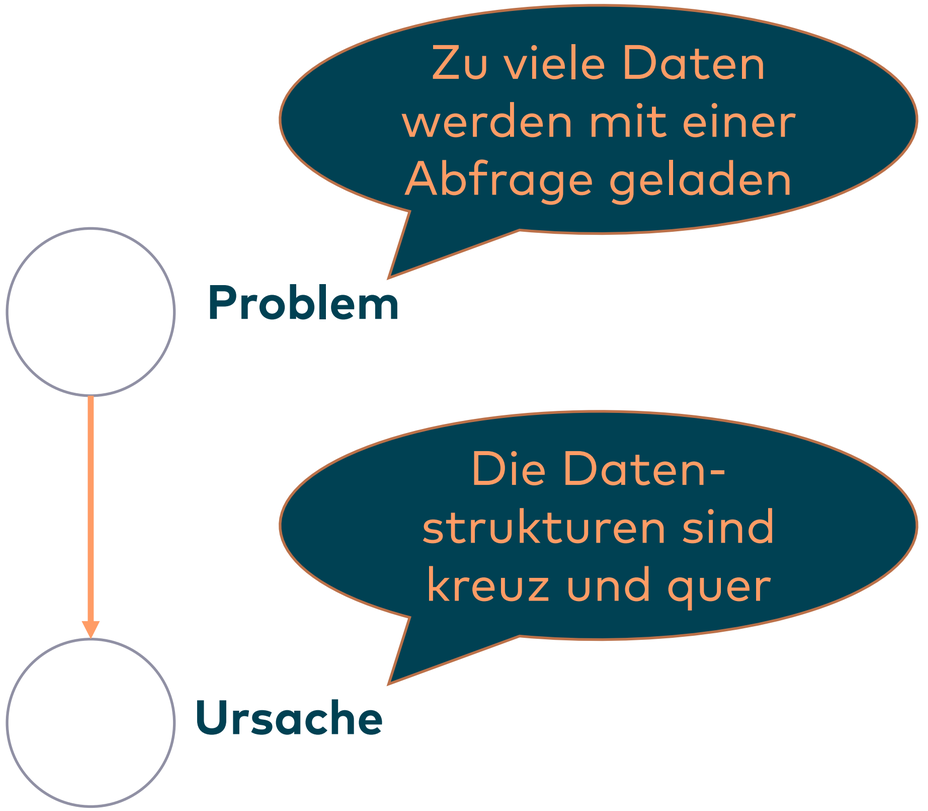
Problem(rück|vor)verschiebung

Problemrückverschiebung

= Root-Cause-Analysis

Frage: Woher kommt das Problem?

→ Der Sache auf den Grund gehen

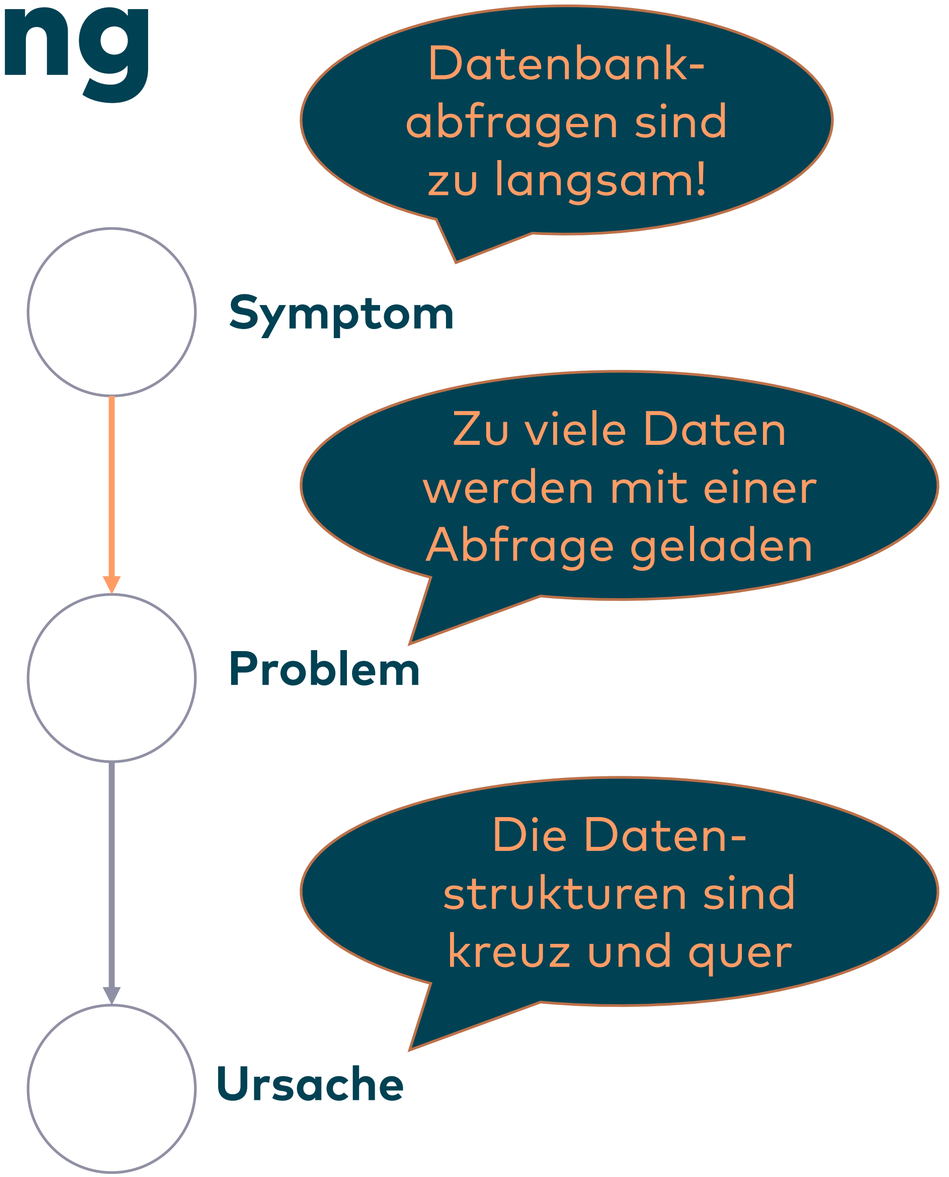


Problemvorverschiebung

= „Reverse-Problem-Analysis“

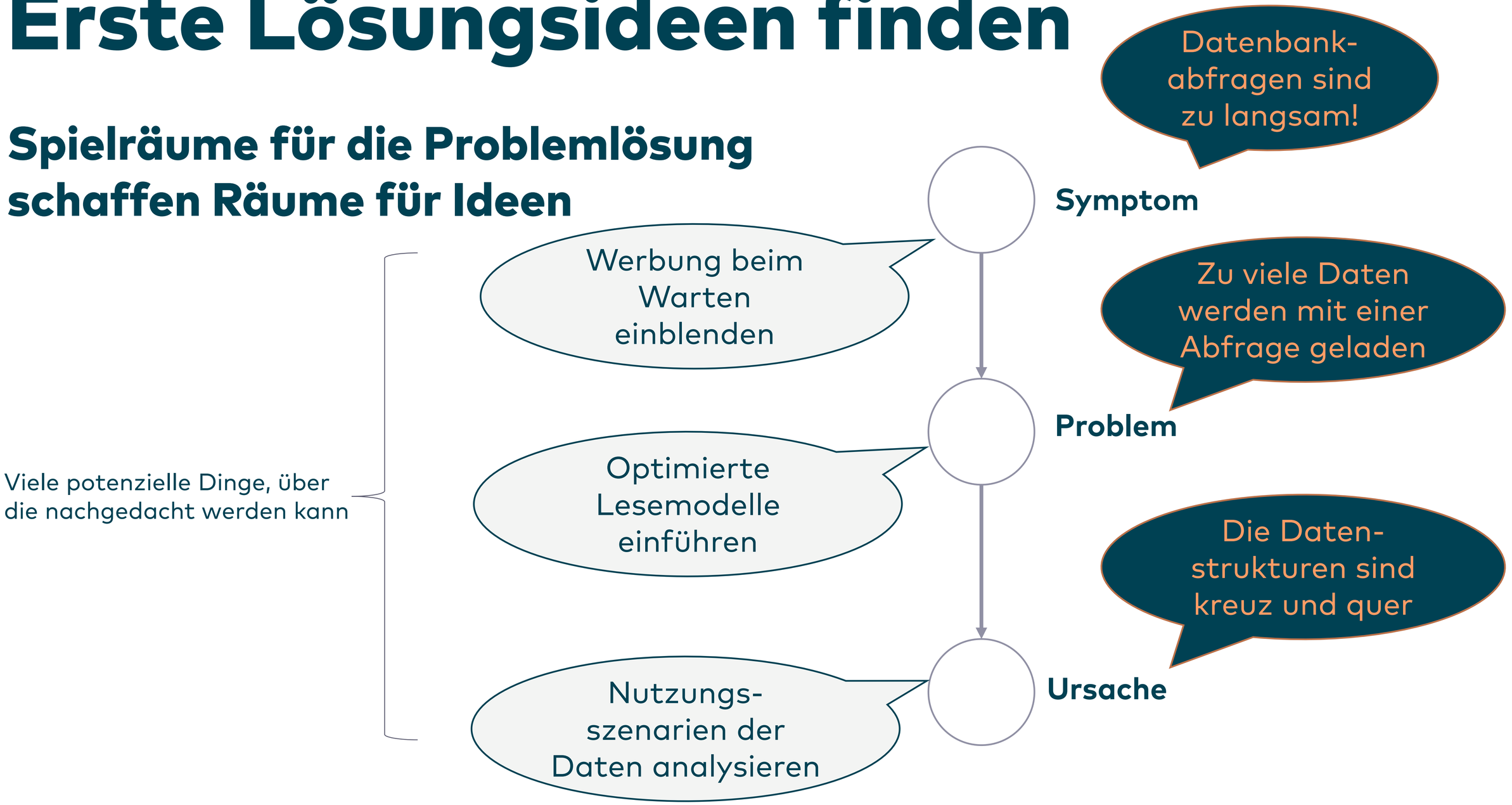
Frage: Wozu führt das Problem?

→ Weitere Einstiegsstellen finden
(aber nur, wenn Lösung nicht anders möglich!)



Erste Lösungsideen finden

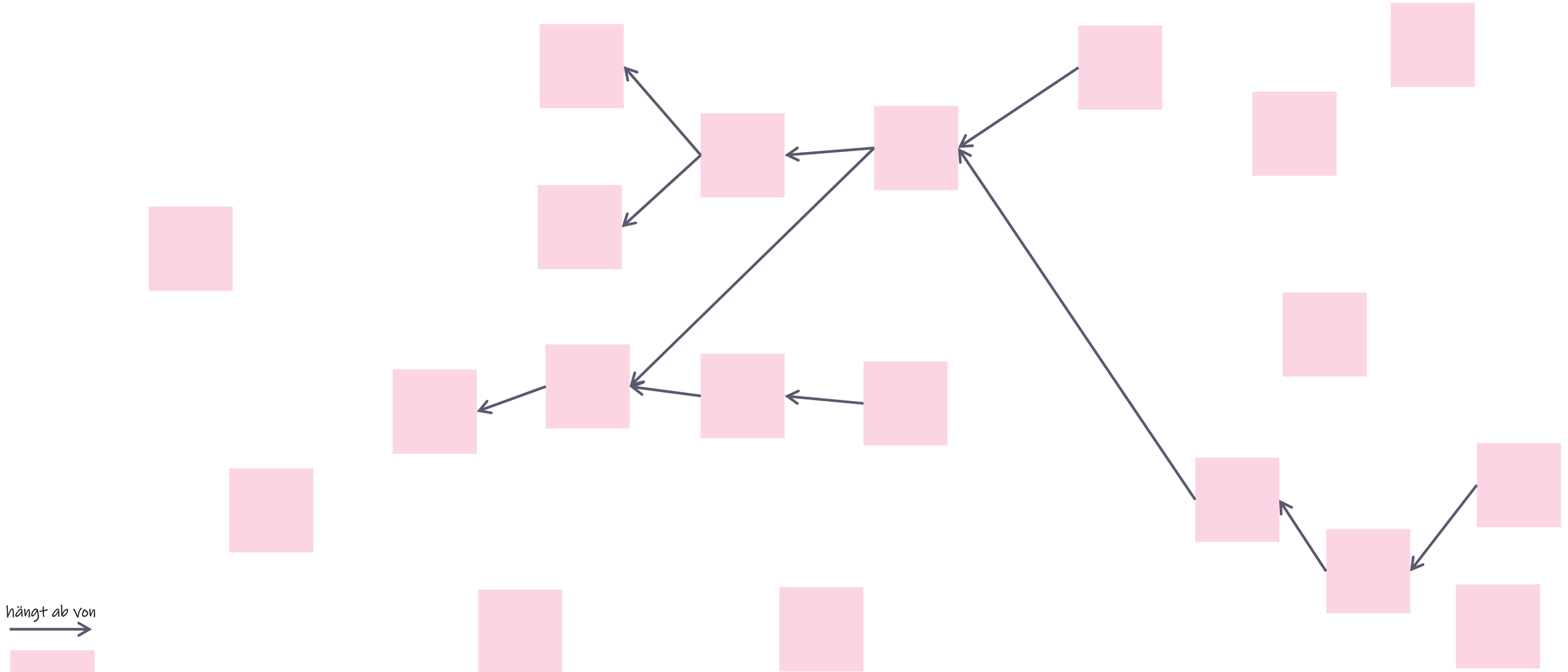
Spielräume für die Problemlösung schaffen Räume für Ideen



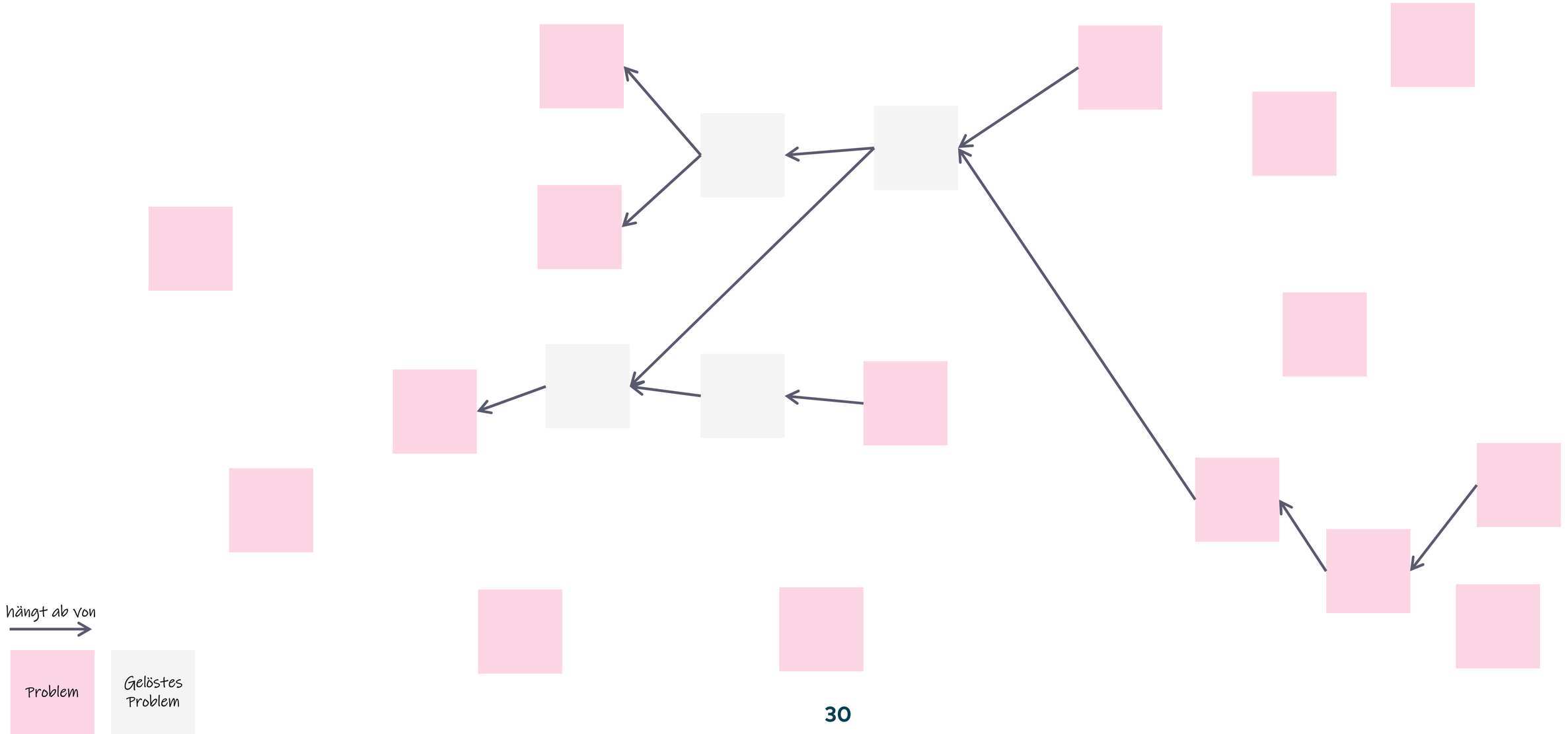
Ergebnisse einer Problemsammlung

Problem

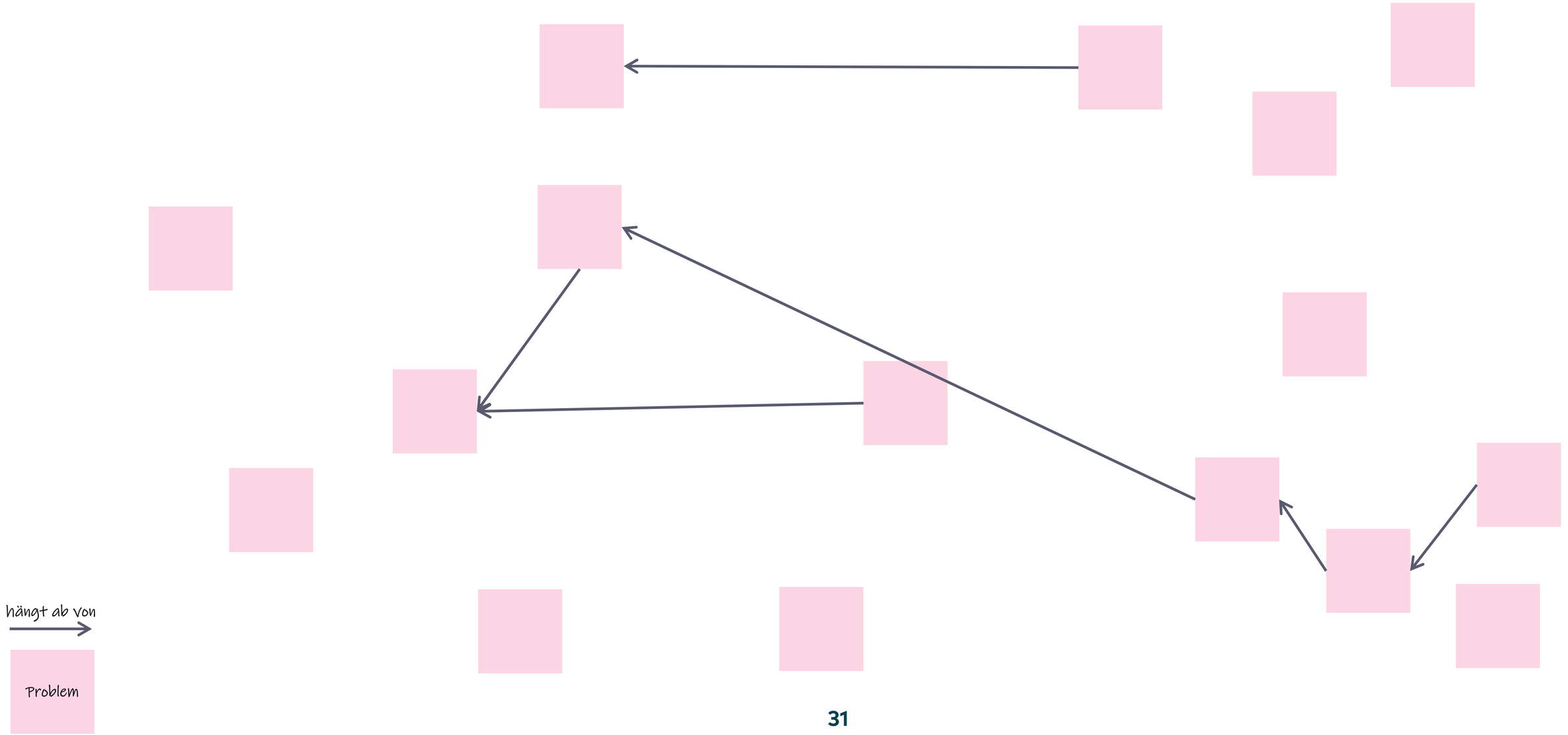
Ergebnisse einer Problemsammlung



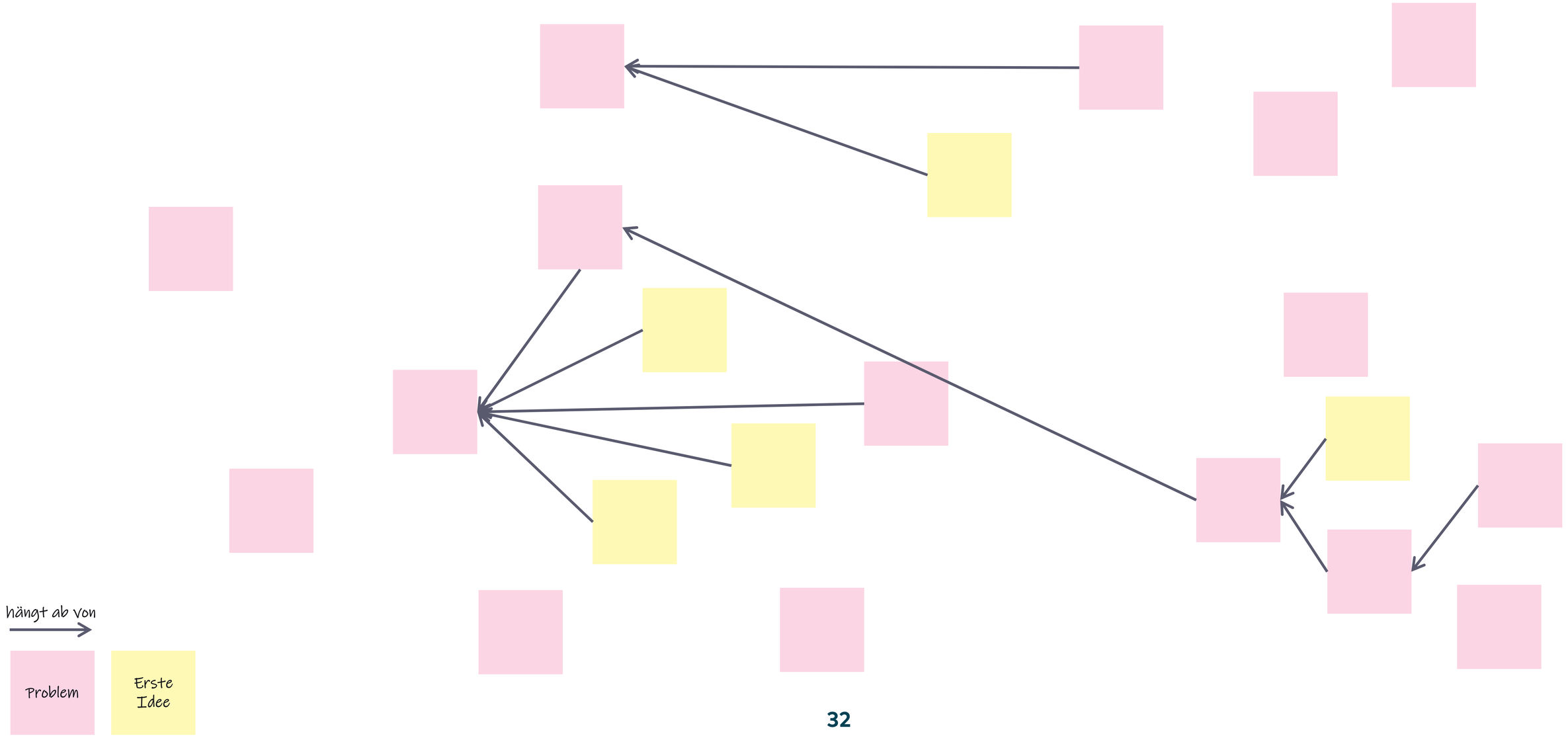
Ergebnisse einer Problemsammlung



Ergebnisse einer Problemsammlung



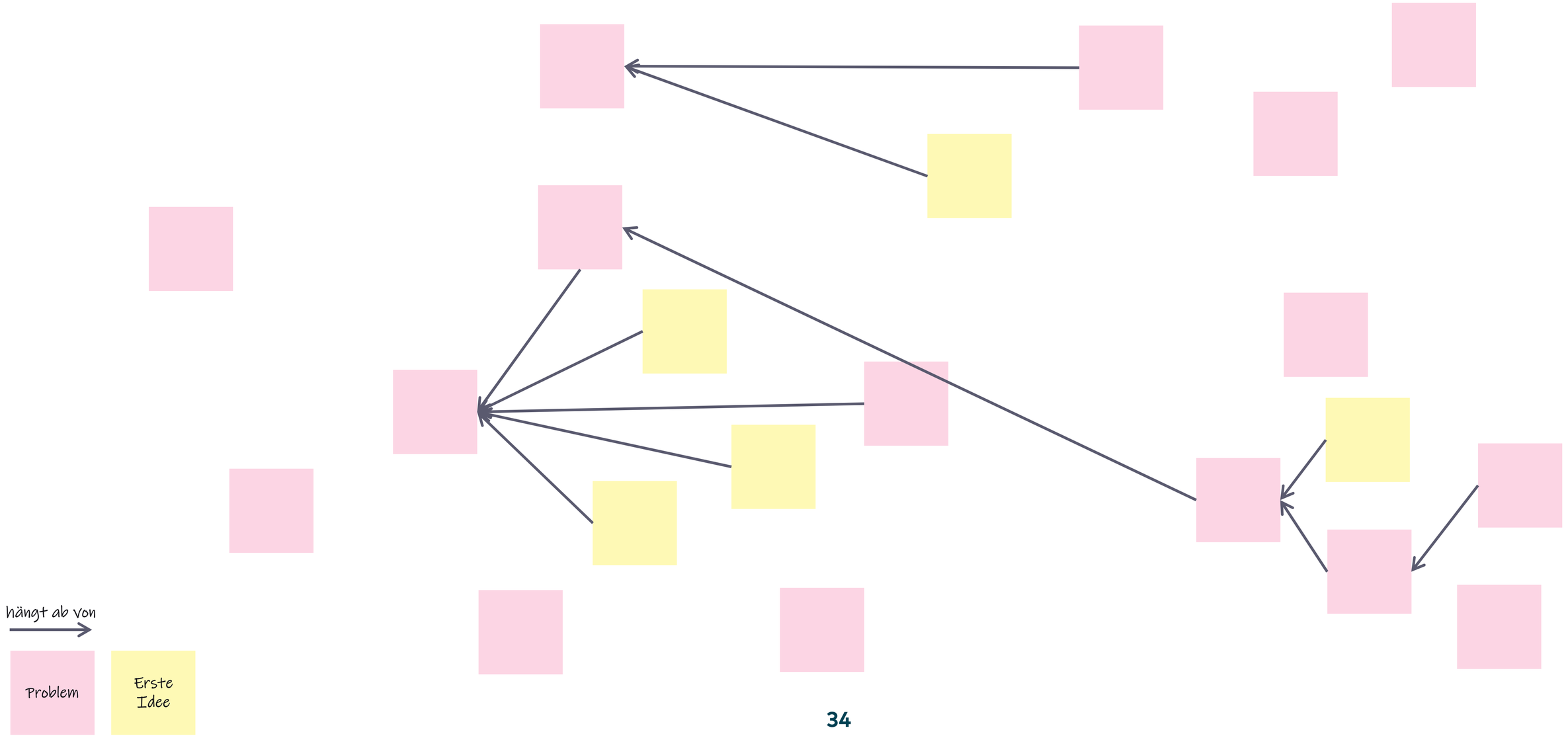
Ergebnisse einer Problemsammlung



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
= Quelle neuer Probleme

Daten-inkonsistenzen

Hohe Betriebs-kosten

Instabile Betriebs-umgebung

Resultat: Verschlimmbesserung

Kommt nicht so gut bei allen an…

Bei Begriffen genauer werden

Antwortzeiten der Anwendung dauern zu lange?

Schlechte Performance

Umsetzung von Features dauert zu lange?

Durchsatz von Vertrags-genehmigungen zu langsam?

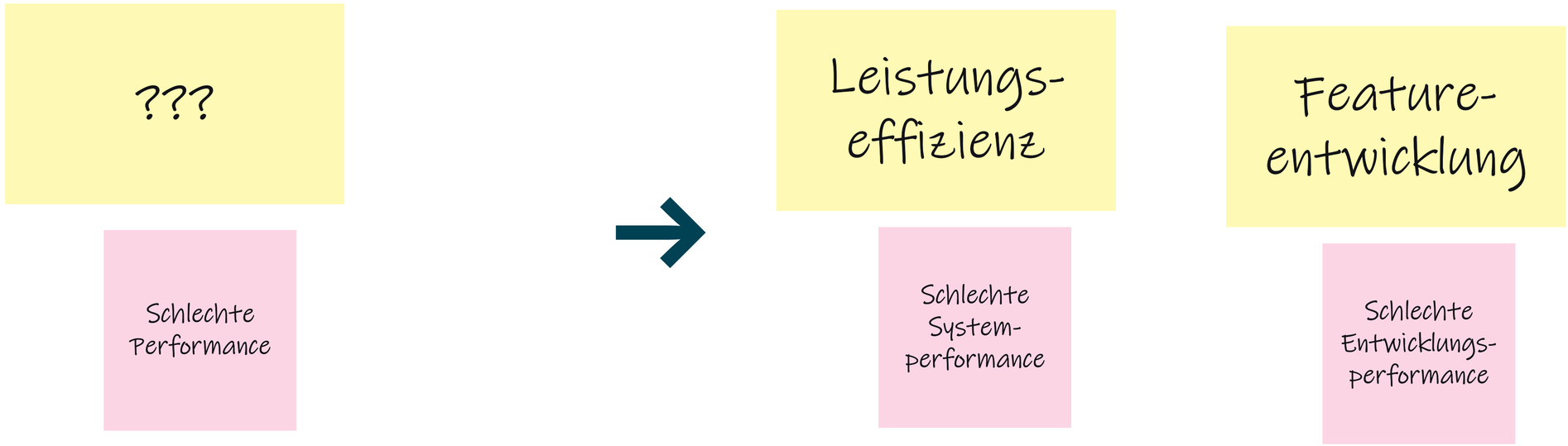
Entwickler tanzen zu schlecht?

„Was meinst du damit?“

Einfache Frage, große Wirkung!

Begriffe im Kontext sehen

Wörter bekommen erst in einem Kontext eine Bedeutung



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

Bei Begriffen 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

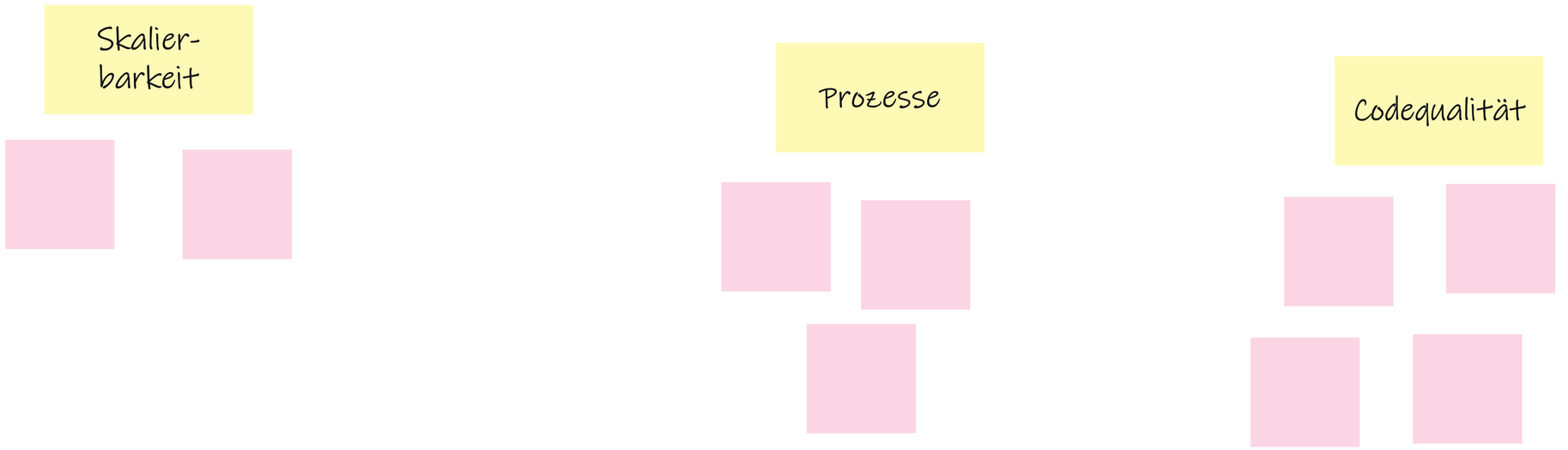
Problem

Ergebnisse einer Problemsammlung

Problem

Cluster

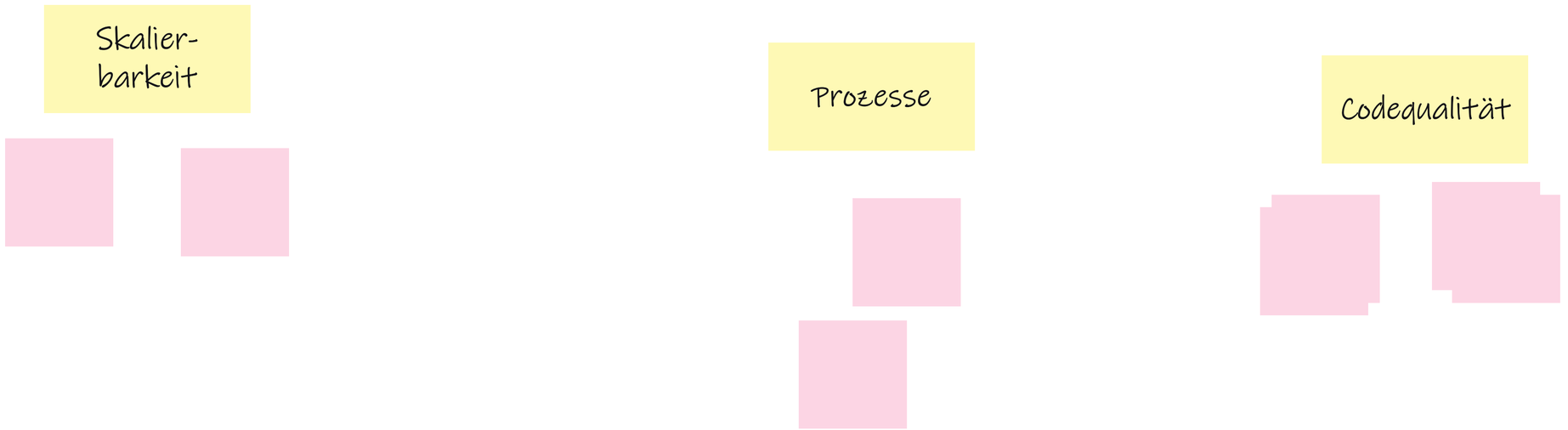
Ergebnisse einer Problemsammlung



Problem

Cluster

Ergebnisse einer Problemsammlung



Problem

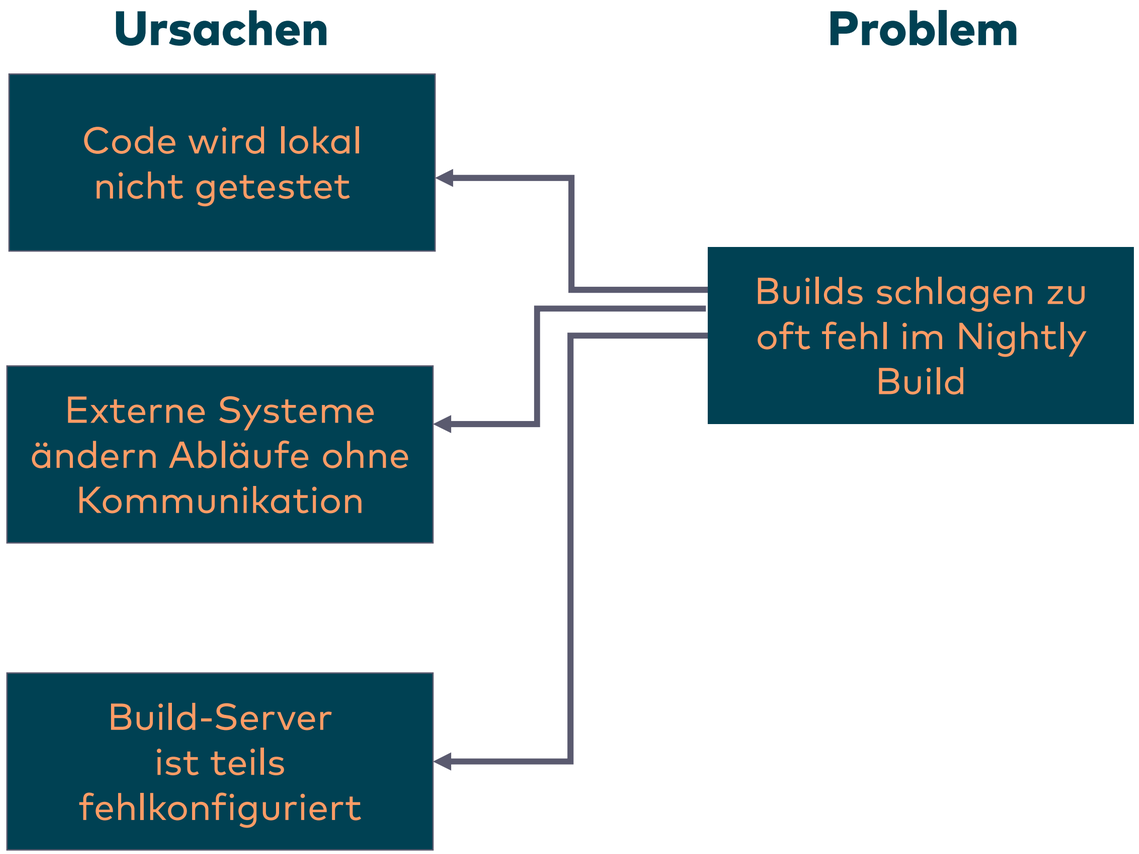
Cluster

4. Punkt

ProblemursachEN

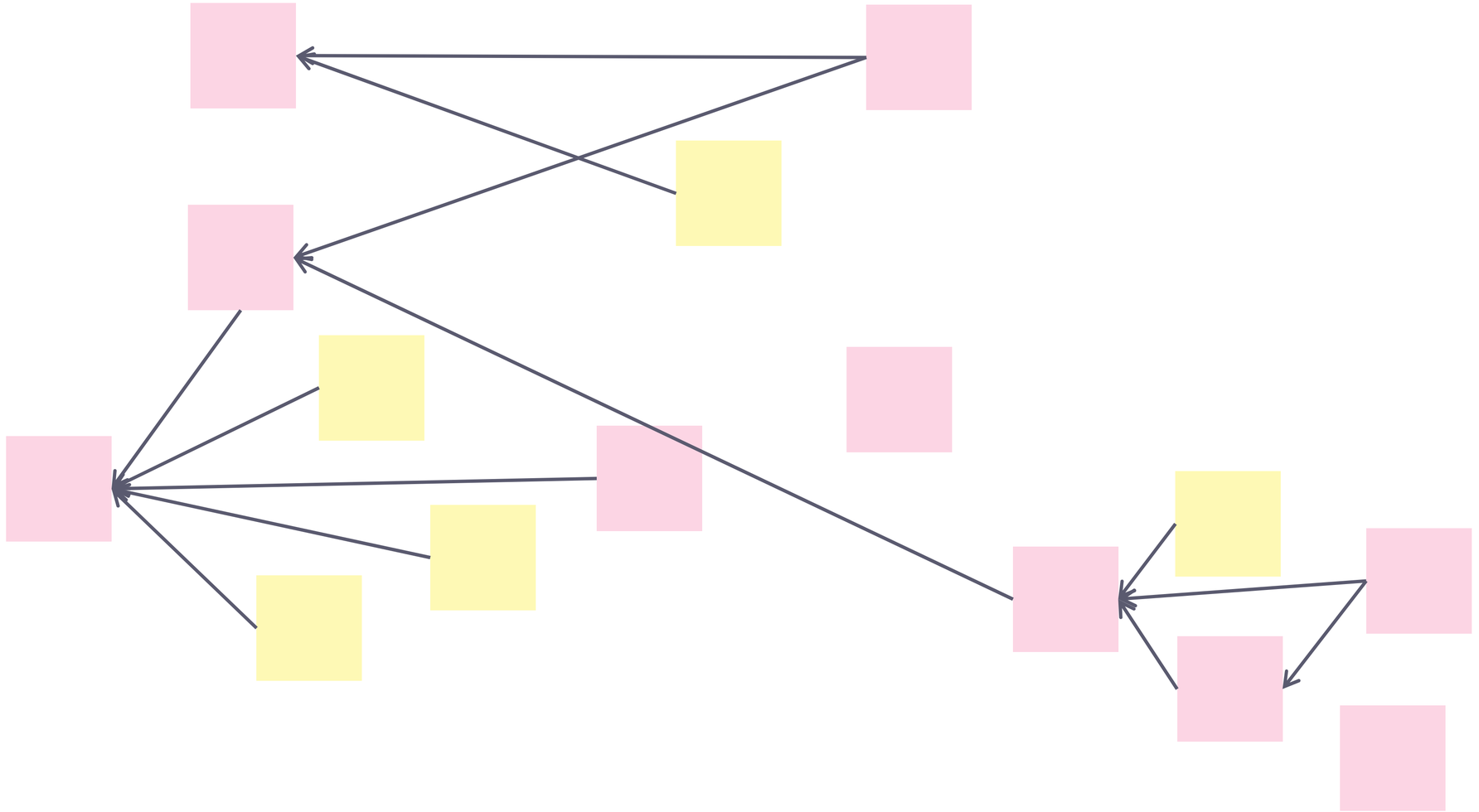
Problem → UrsachEN

UrsachEN betrachten



Wichtige Frage:
Was noch?

Ergebnisse einer Problemsammlung



hängt ab von

Problem

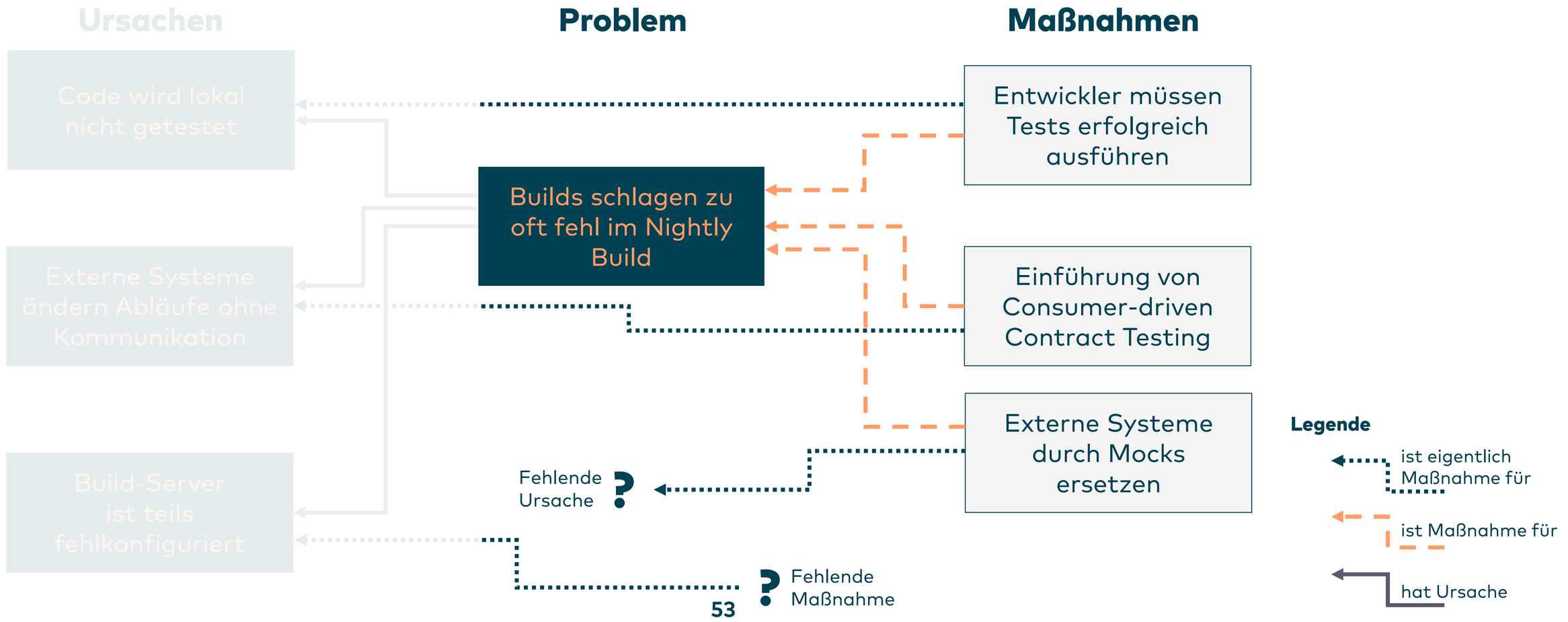
Erste Idee

5. Punkt

Passende Maßnahmen finden

Maßnahmen zu Ursachen finden

Gefahr: Maßnahmen oder Ursachen unvollständig



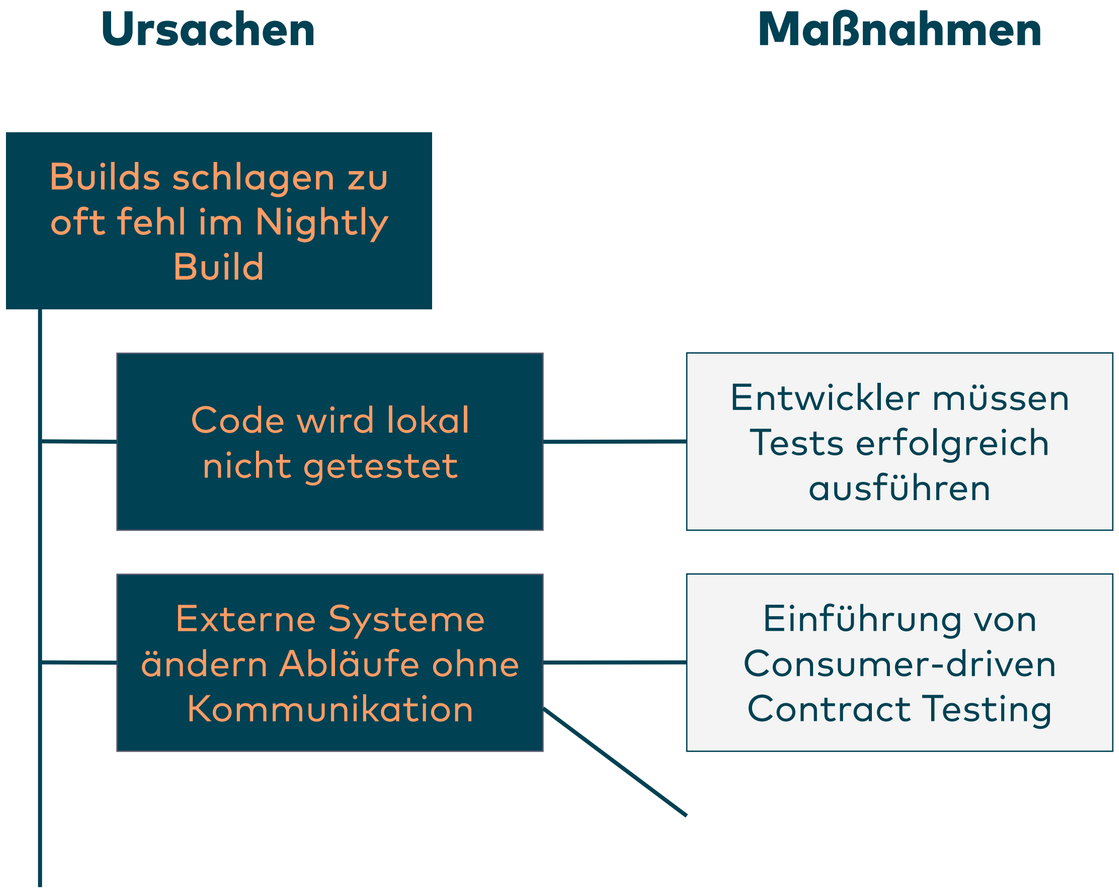
"We fail more often ***because we solve the wrong problem*** than ***because we get the wrong solution to the right problem***."

Russell L. Ackoff

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

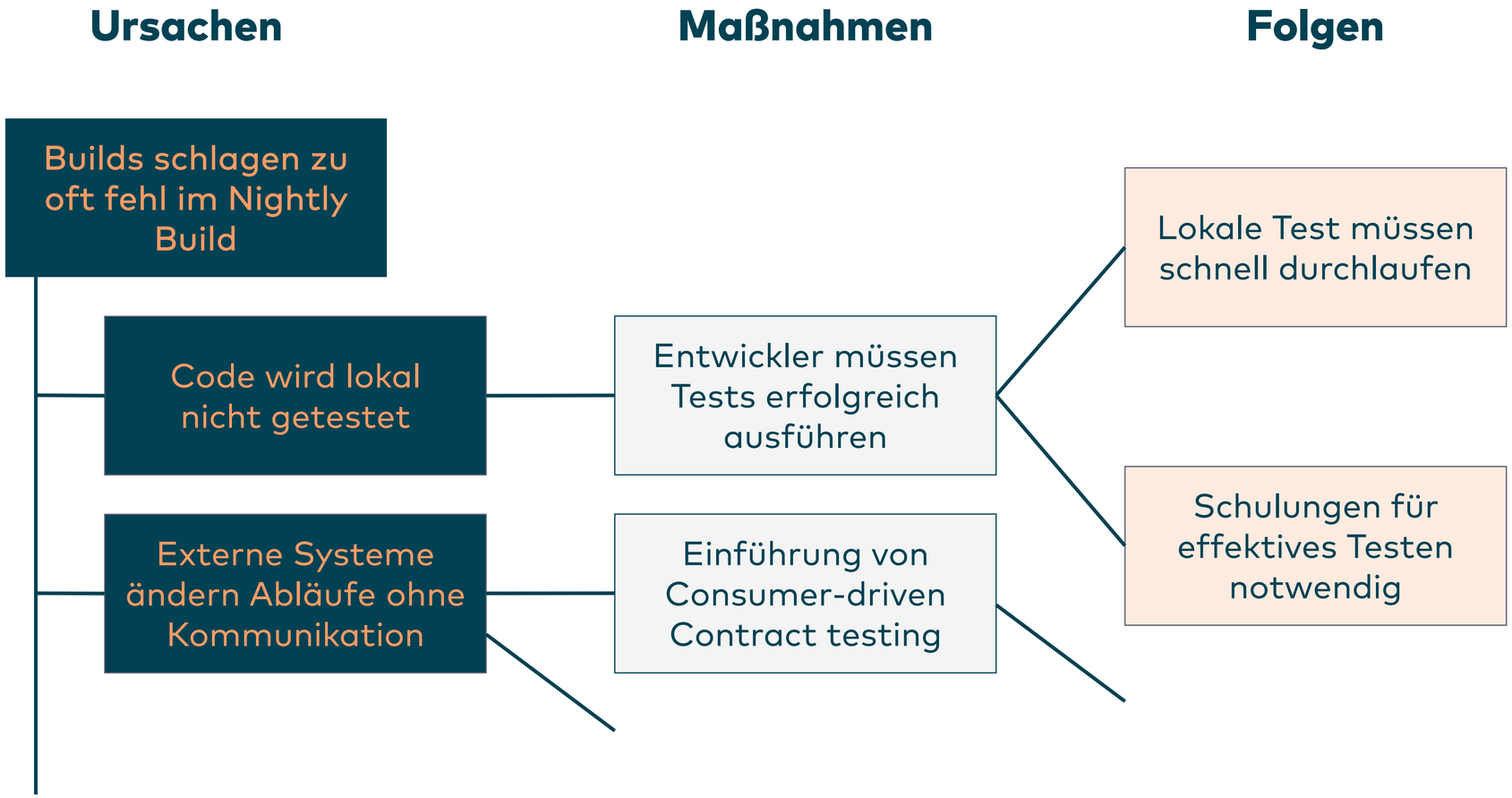
Landkarte der Probleme

Ursachen mit Maßnahmen zusammenbringen



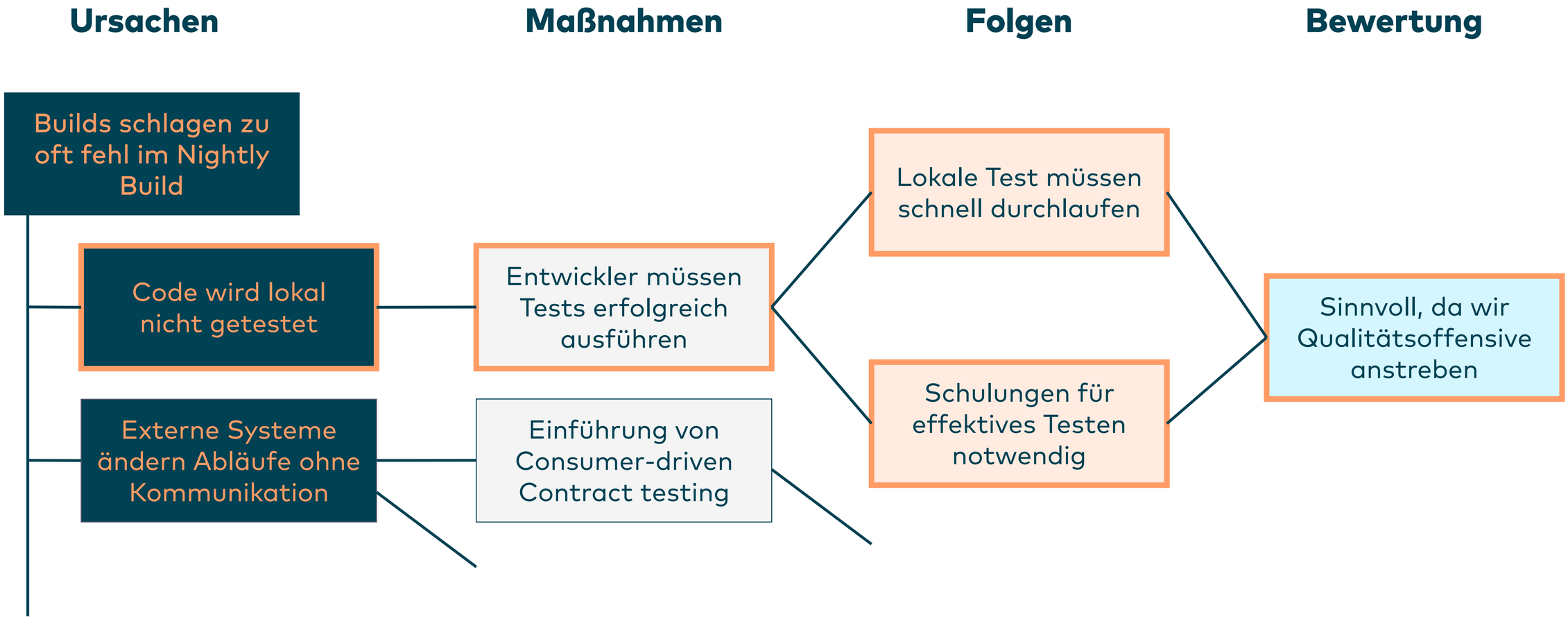
Landkarte der möglichen Lösungen

Folgen der Maßnahmen identifizieren (nichts ist umsonst!)



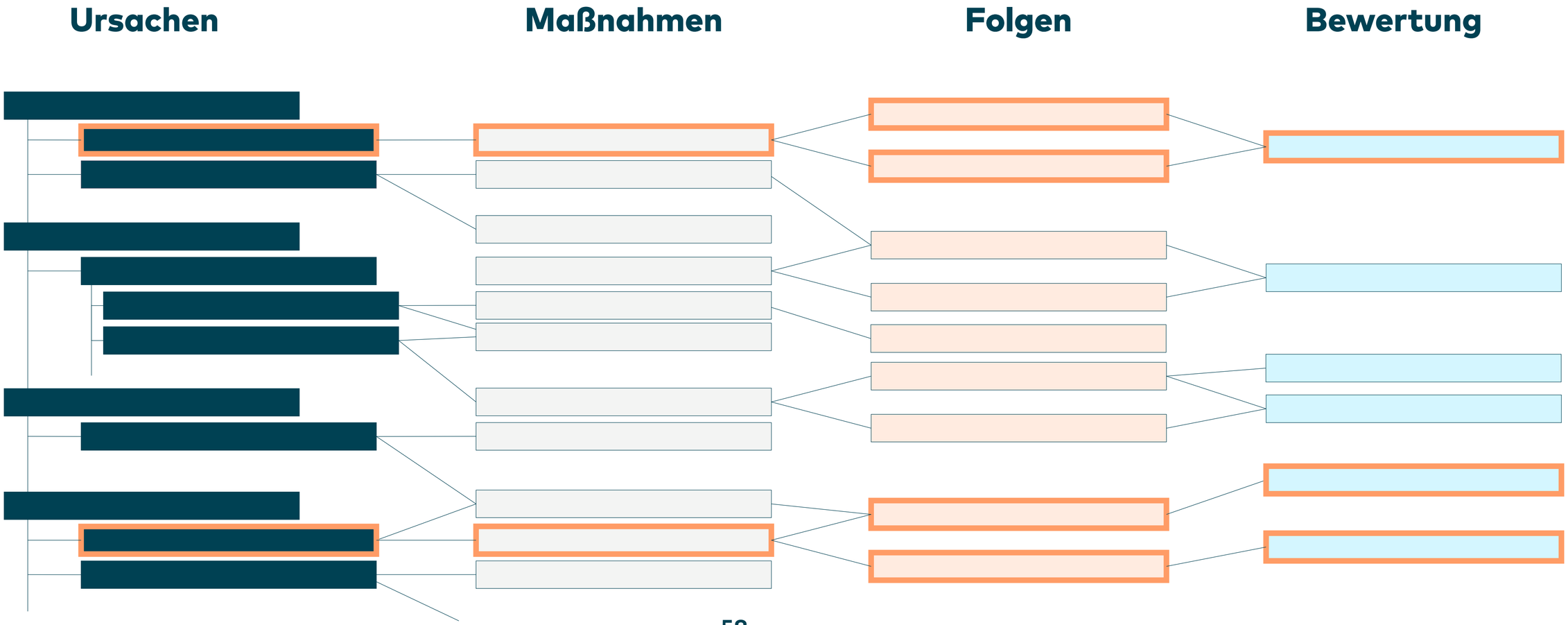
Landkarte der Lösungen

Folgen der Maßnahmen bewerten für die eigene Situation

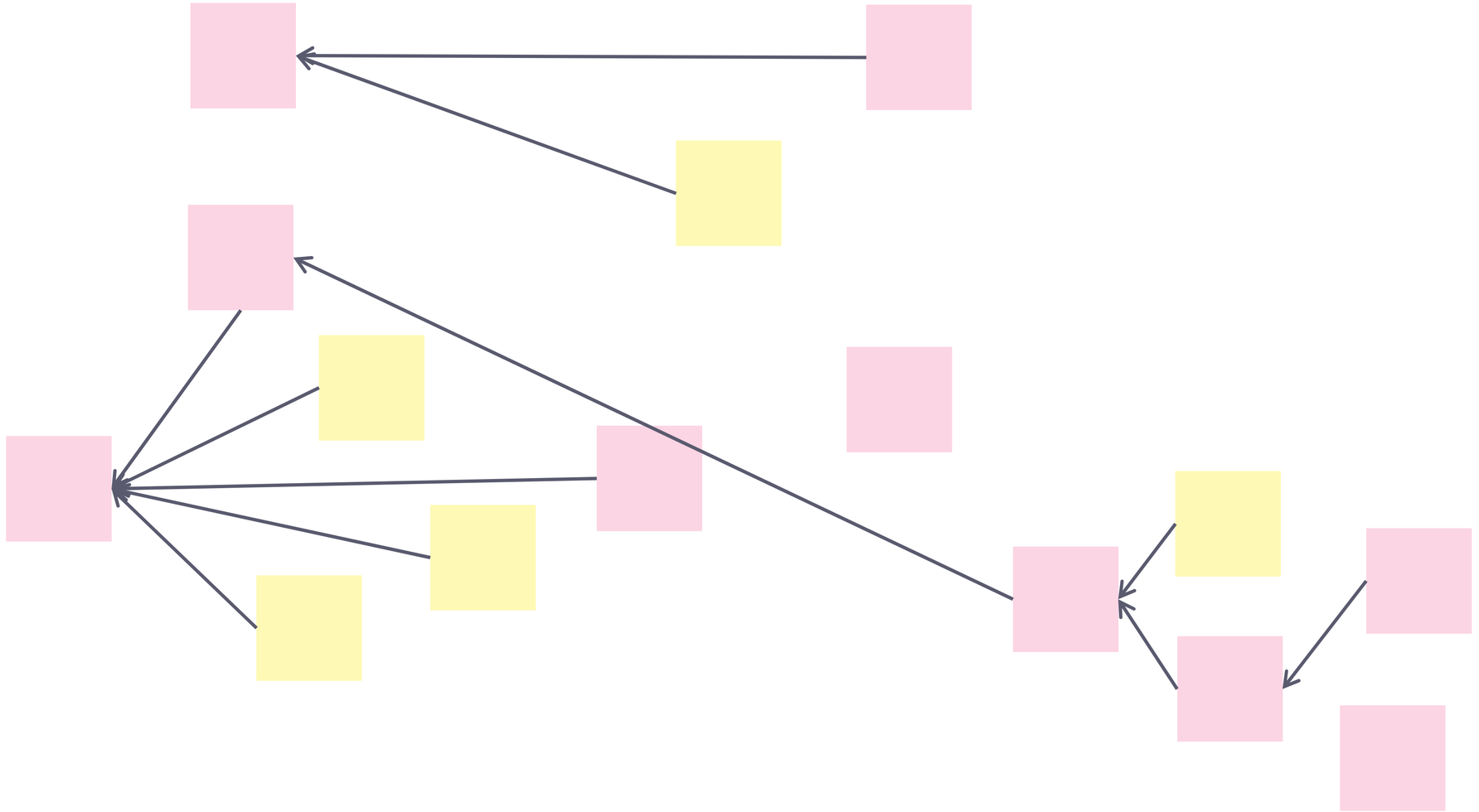


Landkarte der Lösungen

Übersicht über konkrete Maßnahmenbündel schaffen



Ergebnisse einer Problemsammlung

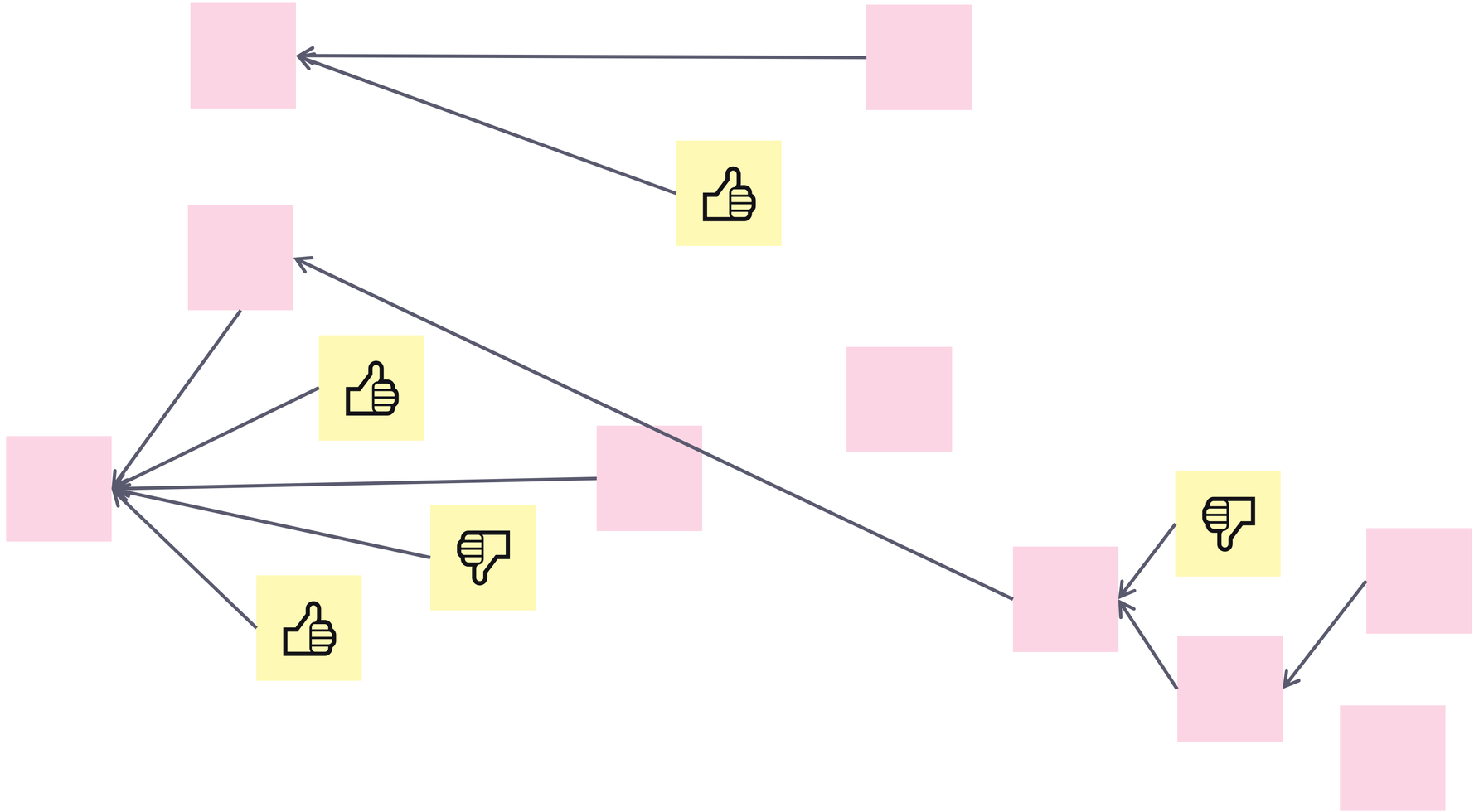


hängt ab von

Problem

Erste Idee

Ergebnisse einer Problemsammlung

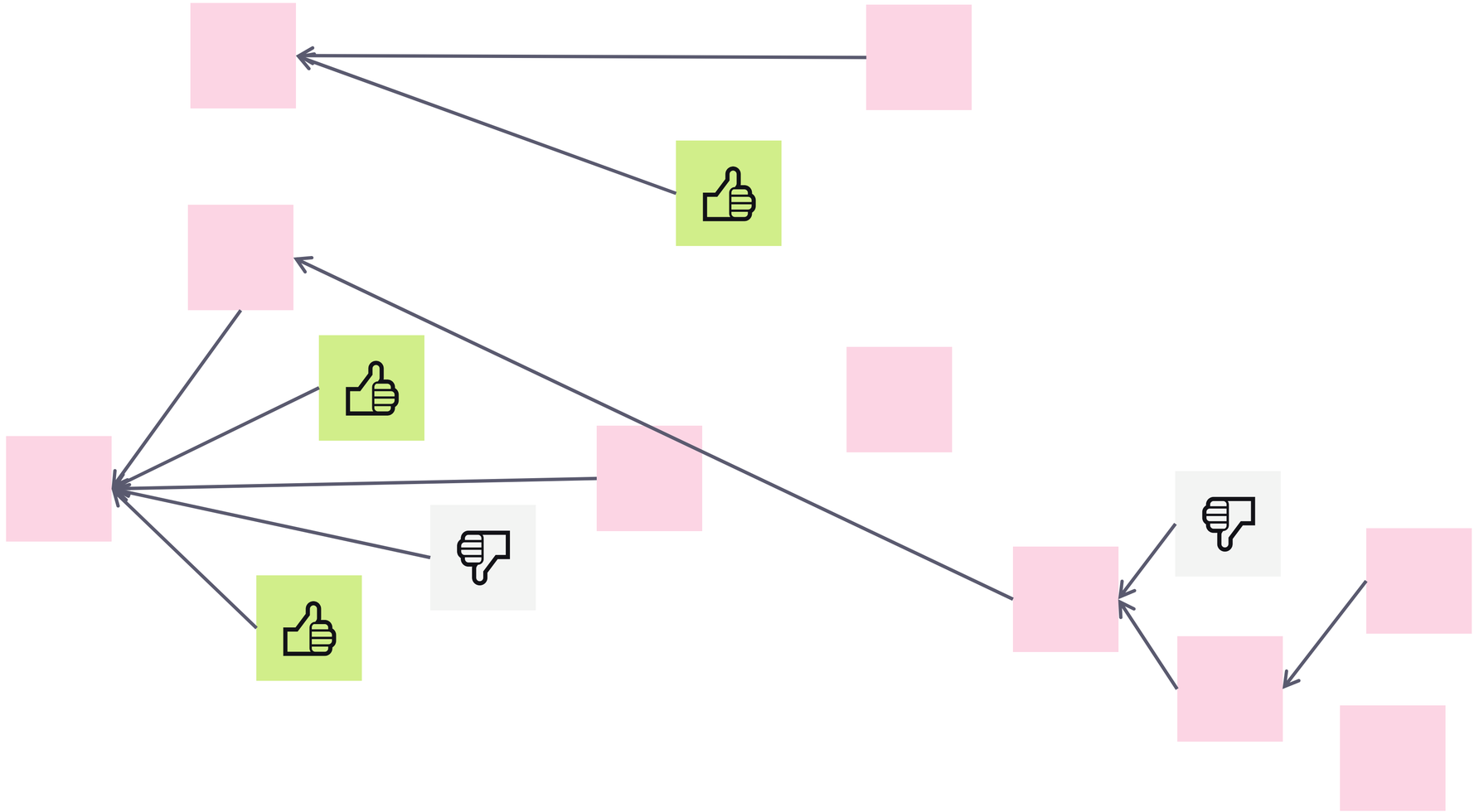


hängt ab von

Problem

Erste Idee

Ergebnisse einer Problemsammlung



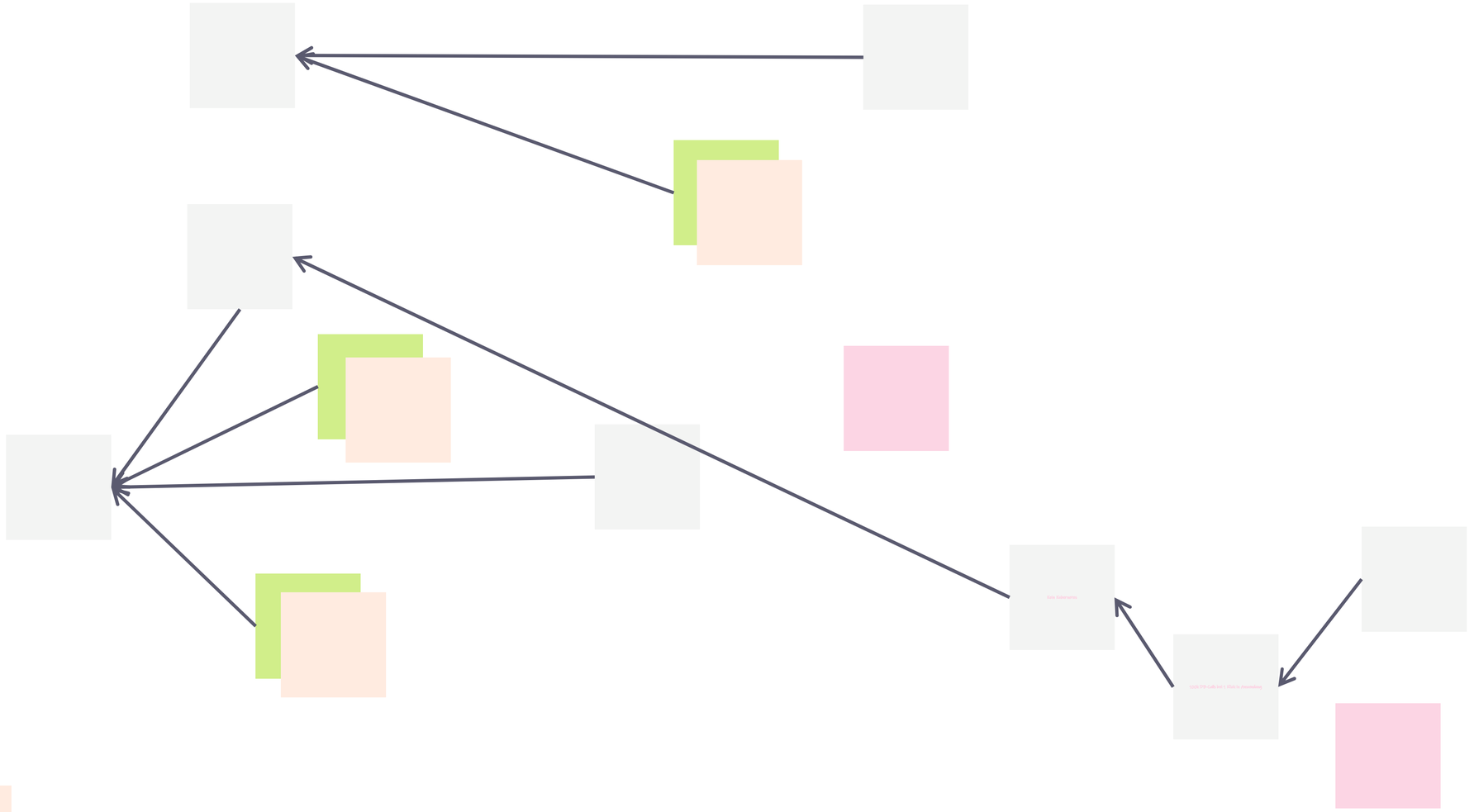
hängt ab von

Problem

Maß-
nahme

Keine
Maß-
nahme

Ergebnisse einer Problemsammlung



hängt ab von

Problem

Maß-
nahme

Gelöstes
Problem

Folge

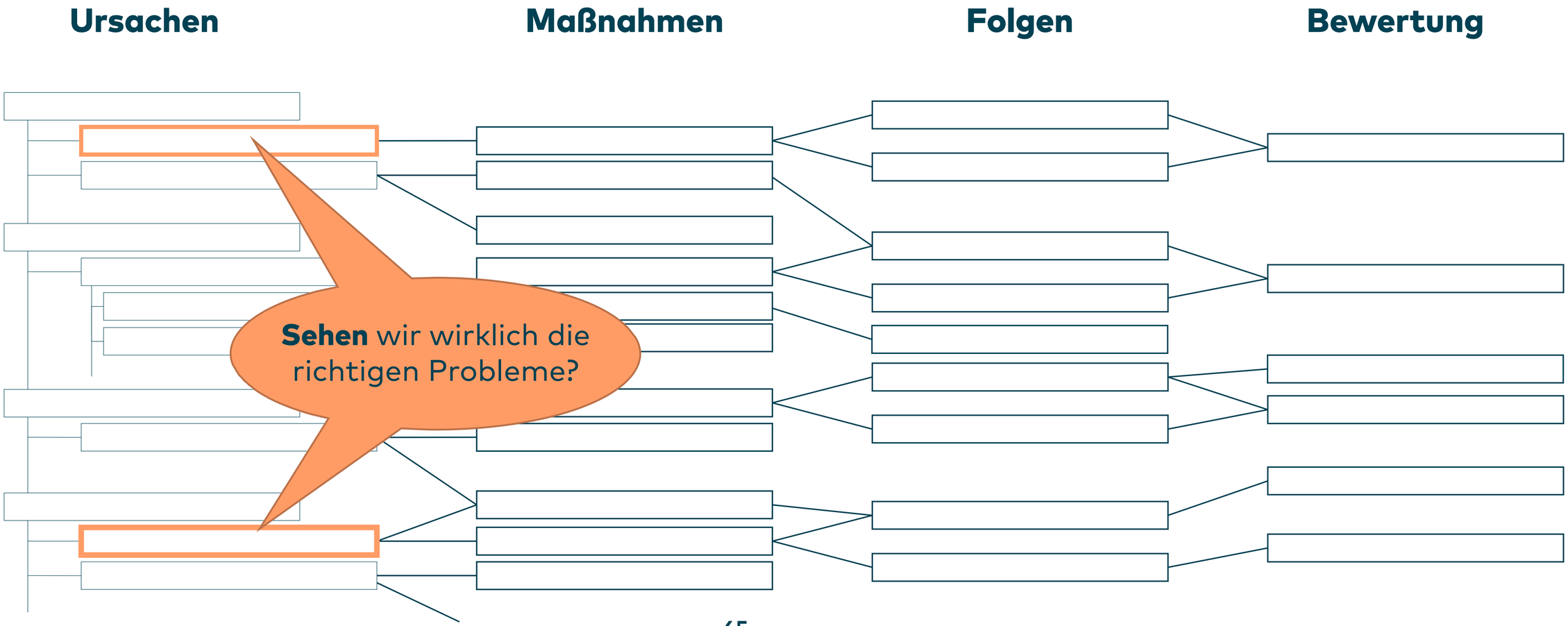
Ergebnisse einer Problemsammlung

6. Punkt

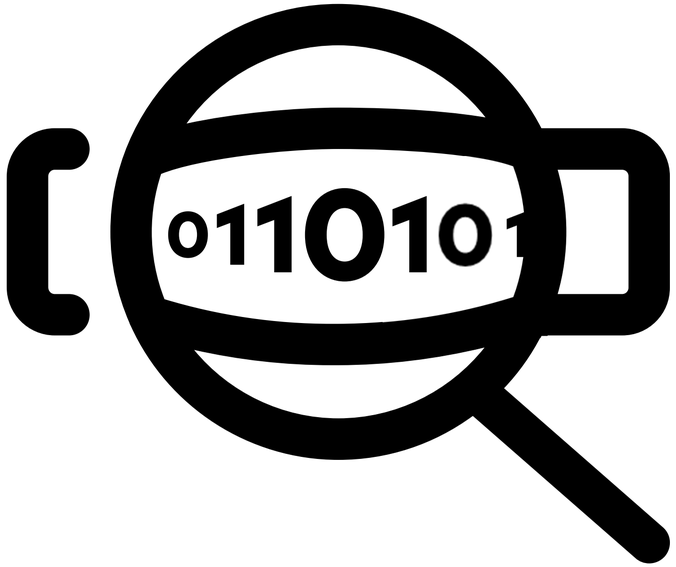
Denkfehler beachten

Sehen wir die wirklichen Probleme?

Problemidentifikation hinterfragen!



Wer nur im Code sucht,

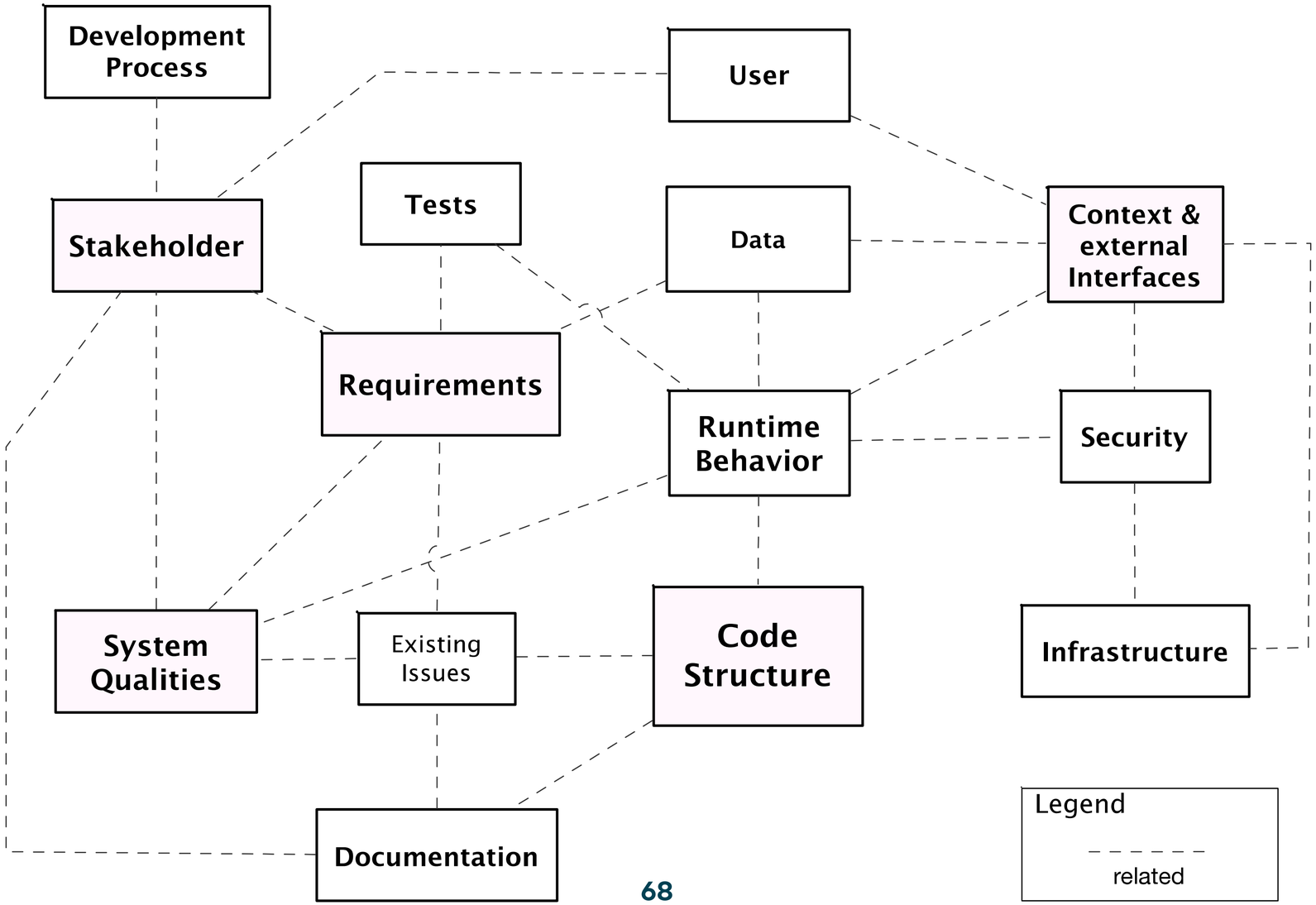


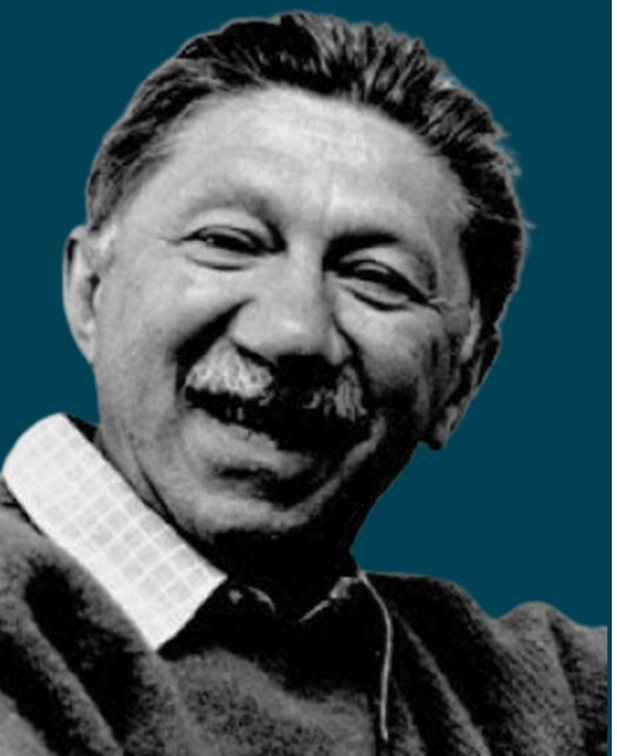
wird auch nur dort Probleme finden!

Antimethode: Den Schlüssel unter der Straßenlaterne suchen

Nicht nur dort suchen,
wo das Licht hinfällt!

Bewusst in die Breite gehen!



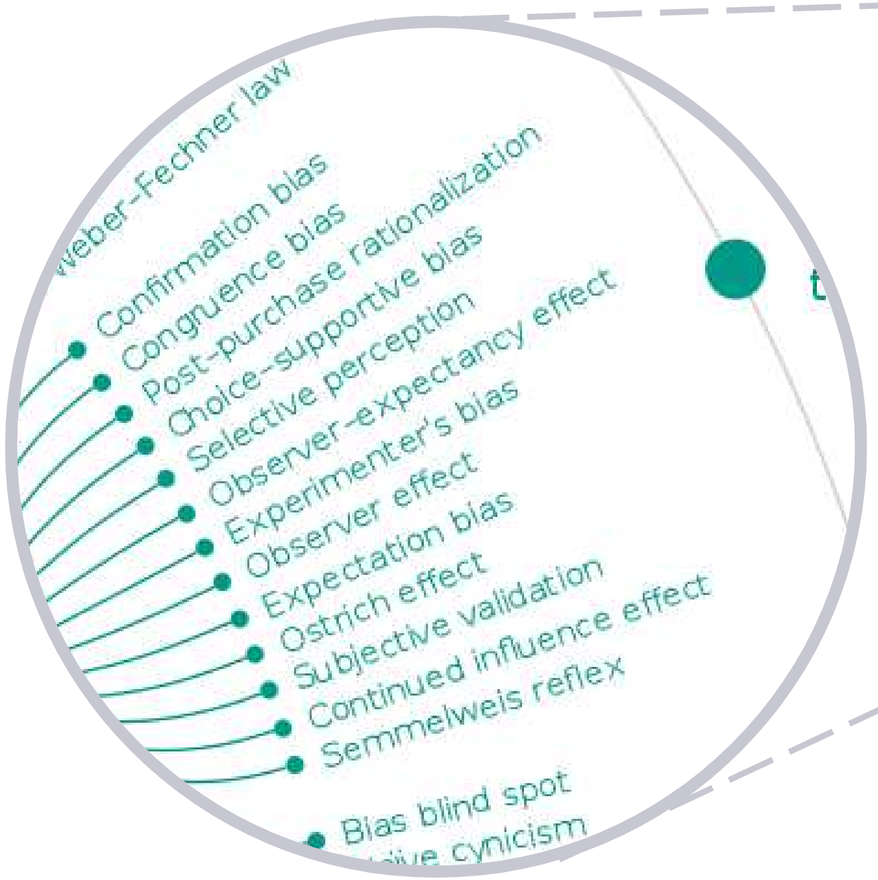


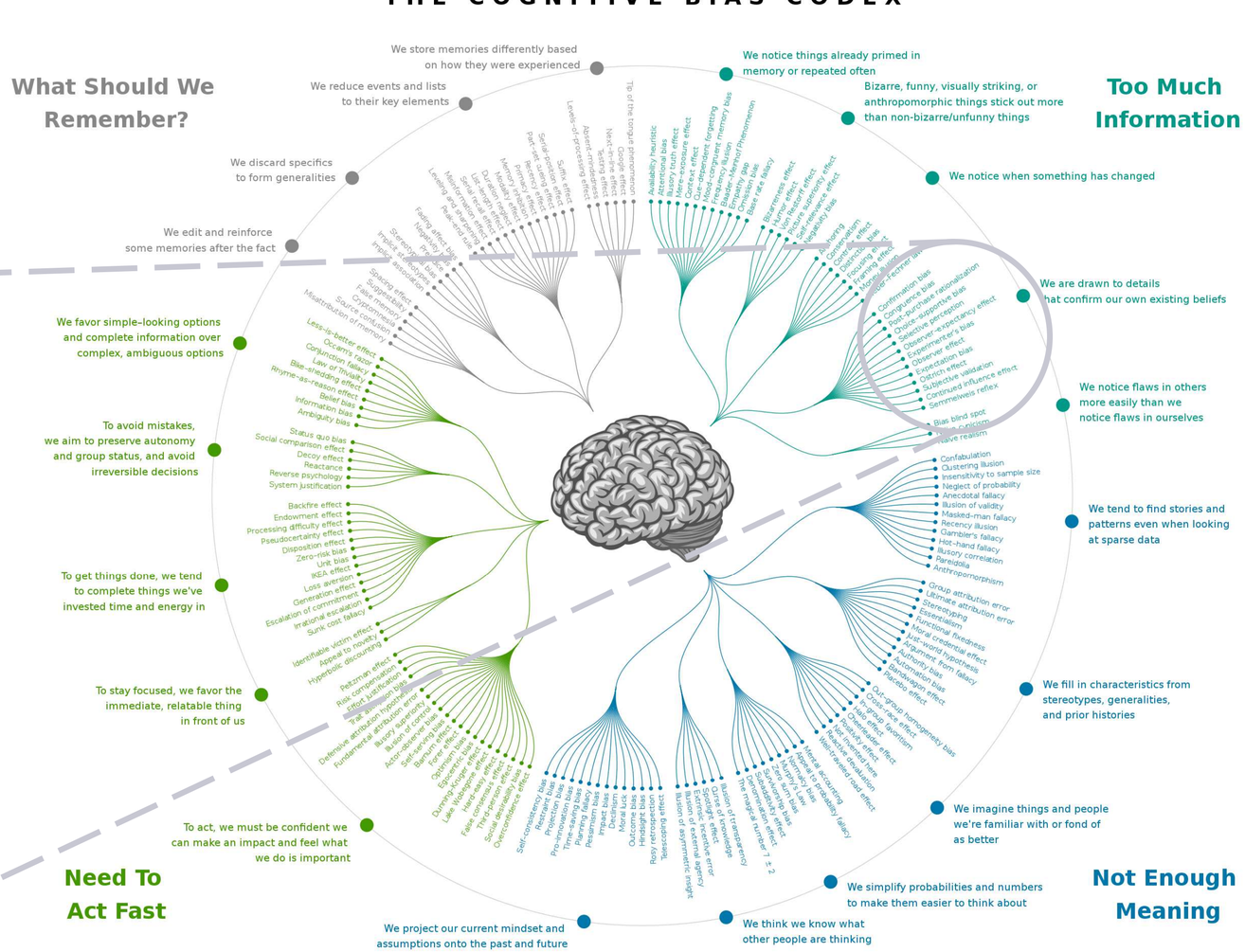
"I suppose it is tempting, if the only tool you have is a hammer, to treat everything as if it were a nail."

Abraham Maslow

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

Die große Welt der Denkfehler





Ein paar Denkfehler als Beispiel

Recency Bias

Emotional Bias

Known Solution Bias

Grand Vision Bias

Sunk Cost Fallacy

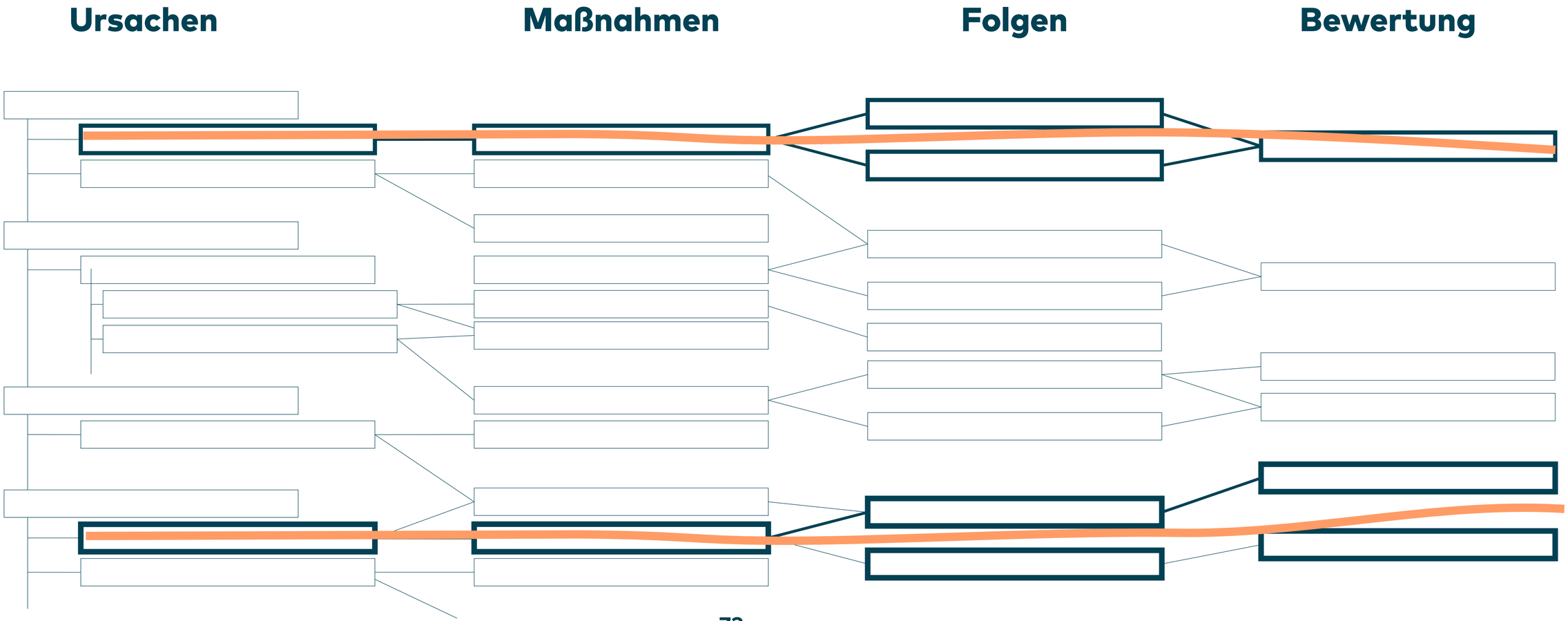
Entnommen von Janelle Arty Starr: Idea Flow - How to Measure the PAIN in Software Development

7. Punkt

Zielführende Problemlösung

Ist unser Weg der richtige?

Problemlösungsvorgehen reflektieren!



Kausale Ketten 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

„Problem… …oder Chance?“

Online-Lottoanbieter

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

!

Miese Skalierbarkeit!

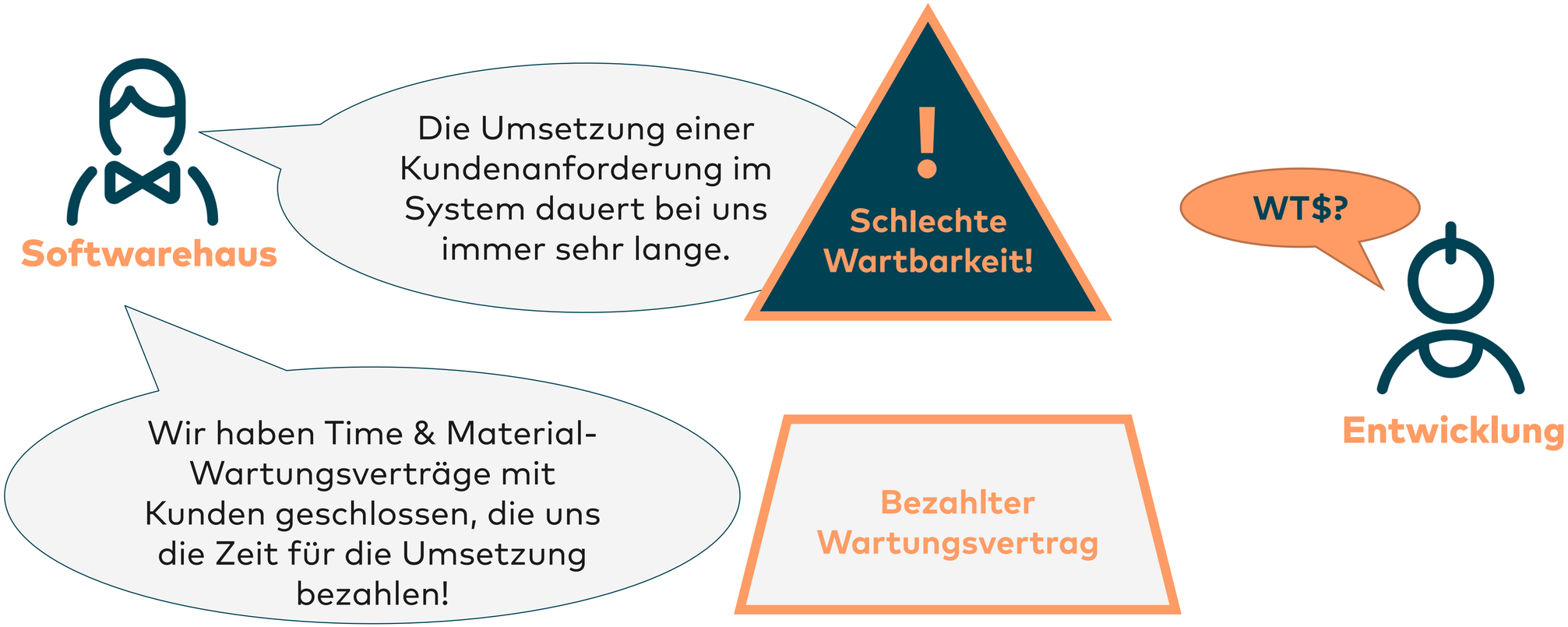
WT$?

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?“

" 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**!

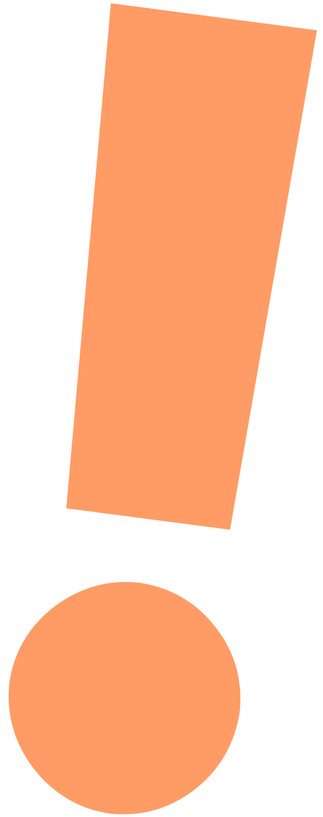
[..]

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

Vielen Dank



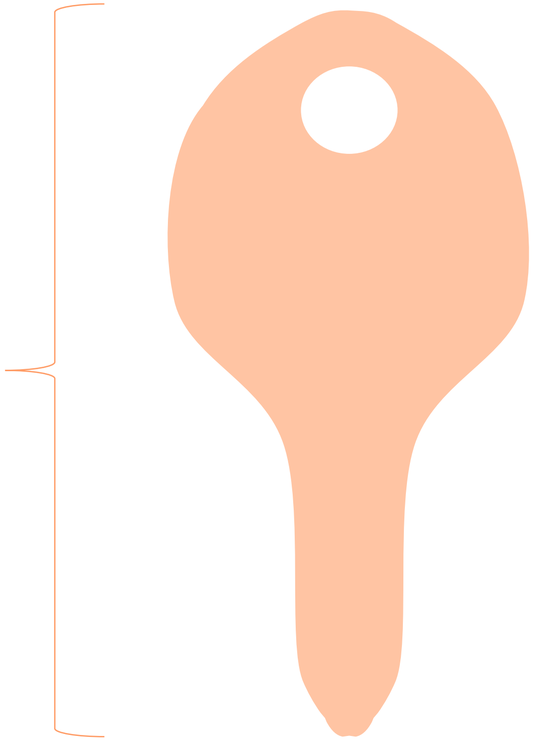
Zusammenfassung

Knifflige Probleme in Softwaresystemen lösen

Wiederholung: Sieben Themen

Problem lösen frei nach Prof. Walter Schönwandts „Key Seven“

Ungefähres Verhältnis der Zeitdauer der Themen



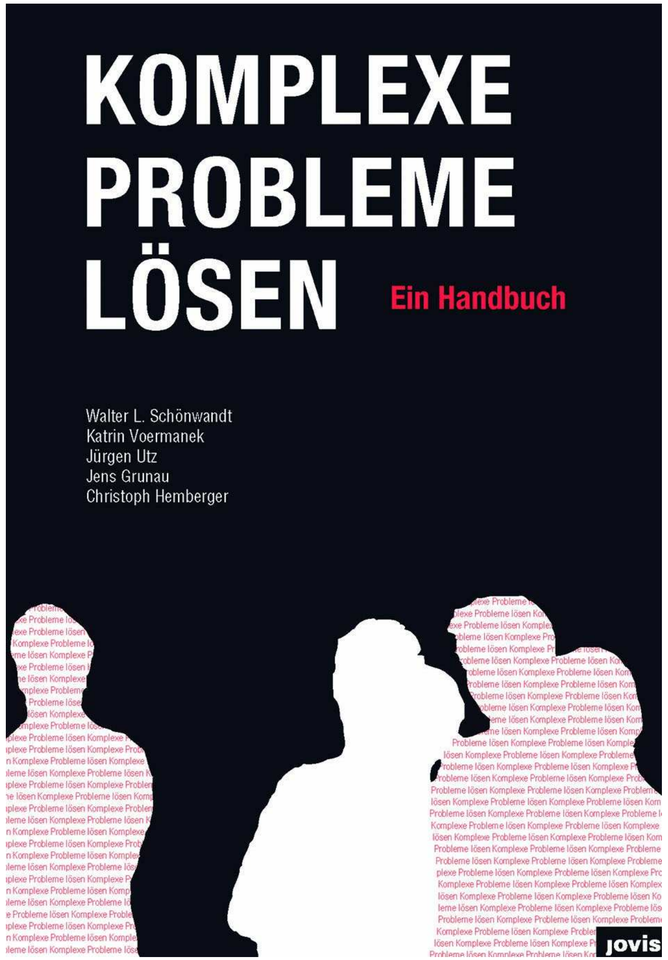
1. Was sind die Probleme?
2. Problem(rück|vor)verschiebung
3. Problemverständnis
4. ProblemursachEN
5. Passende Maßnahmen finden
6. Denkfehler beachten
7. Richtiges Problem?

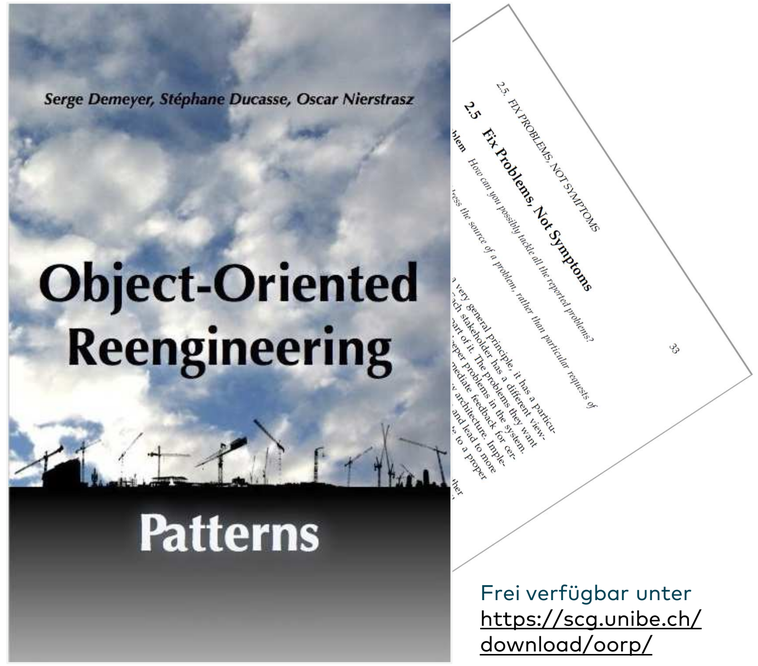
Kudos

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

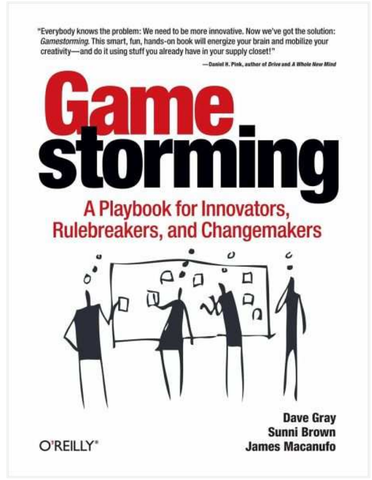
Podcast-Folge von Georg Jocham
Abenteuer Probleme lösen, Folge 021: So löst man komplexe Probleme, im Gespräch mit Prof. Walter Schönwandt
https://georgjocham.com/apl-020-so-loest-man-komplexe-probleme-im-gespraech-mit-prof-dr-walter-schoenwandt/

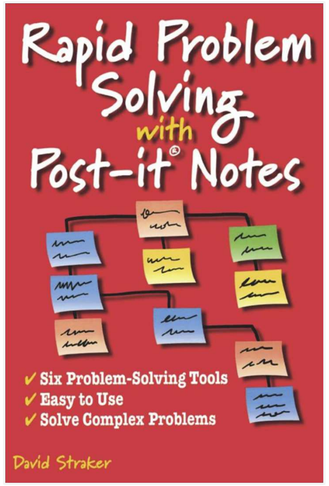
game changer





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





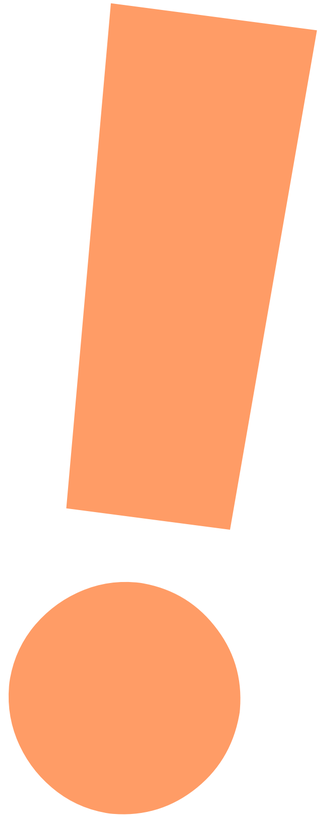
usual suspects

* es gibt aber genügend Berichte über meine Praxiserfahrungen damit!

Fragen

+ Antworten

Vielen Dank



Bis gleich beim INNOQ-Stand?

INNOQ

cards42.org

Cards for Analyzing and Reflecting on Doomed Software

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 86 mobile.de SACAC VORWERK