WIE MAN‘S MACHT, MACHT MAN‘S FALSCH! SOFTWAREQUALITÄT... UND WORAUF ES DABEI ANKOMMT

Stephan Pirnbaum

Moritz Pflügner

07.12.2023

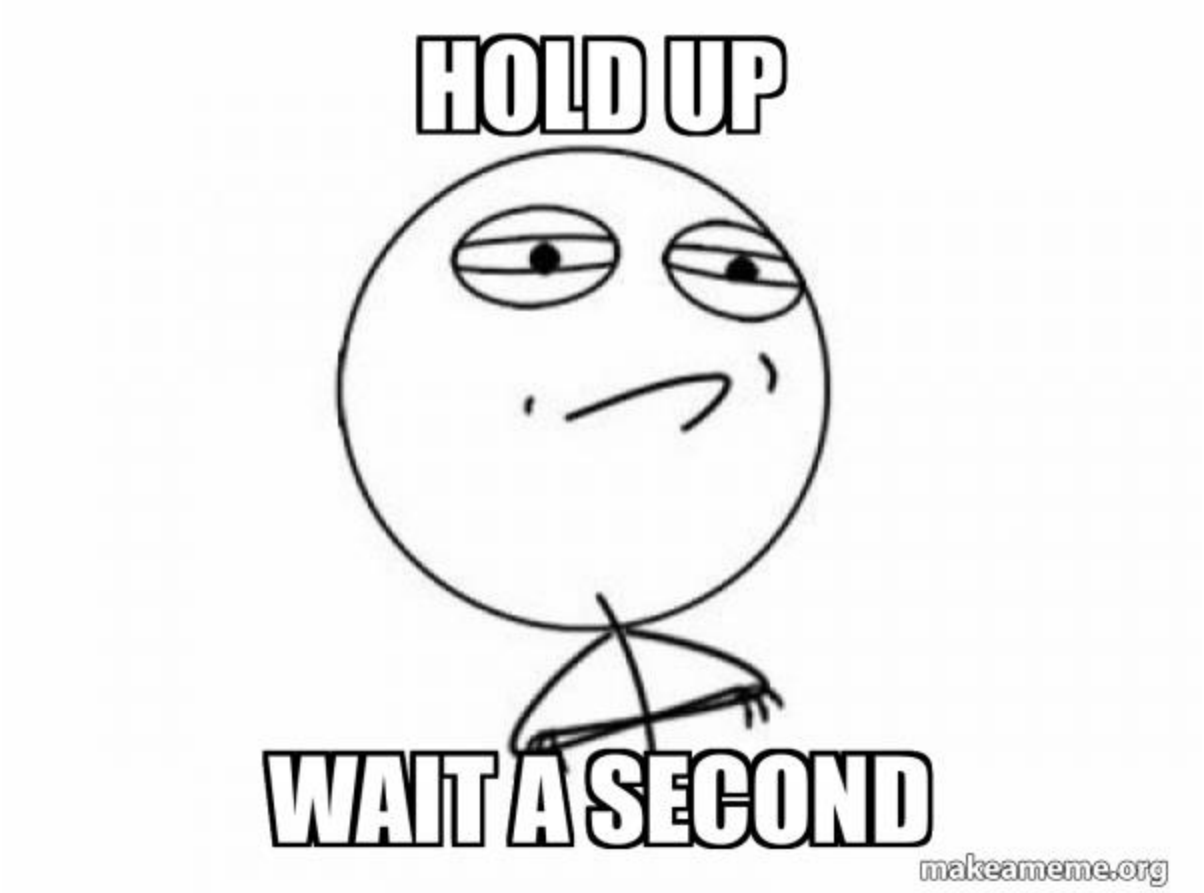
BUSCHMAIS

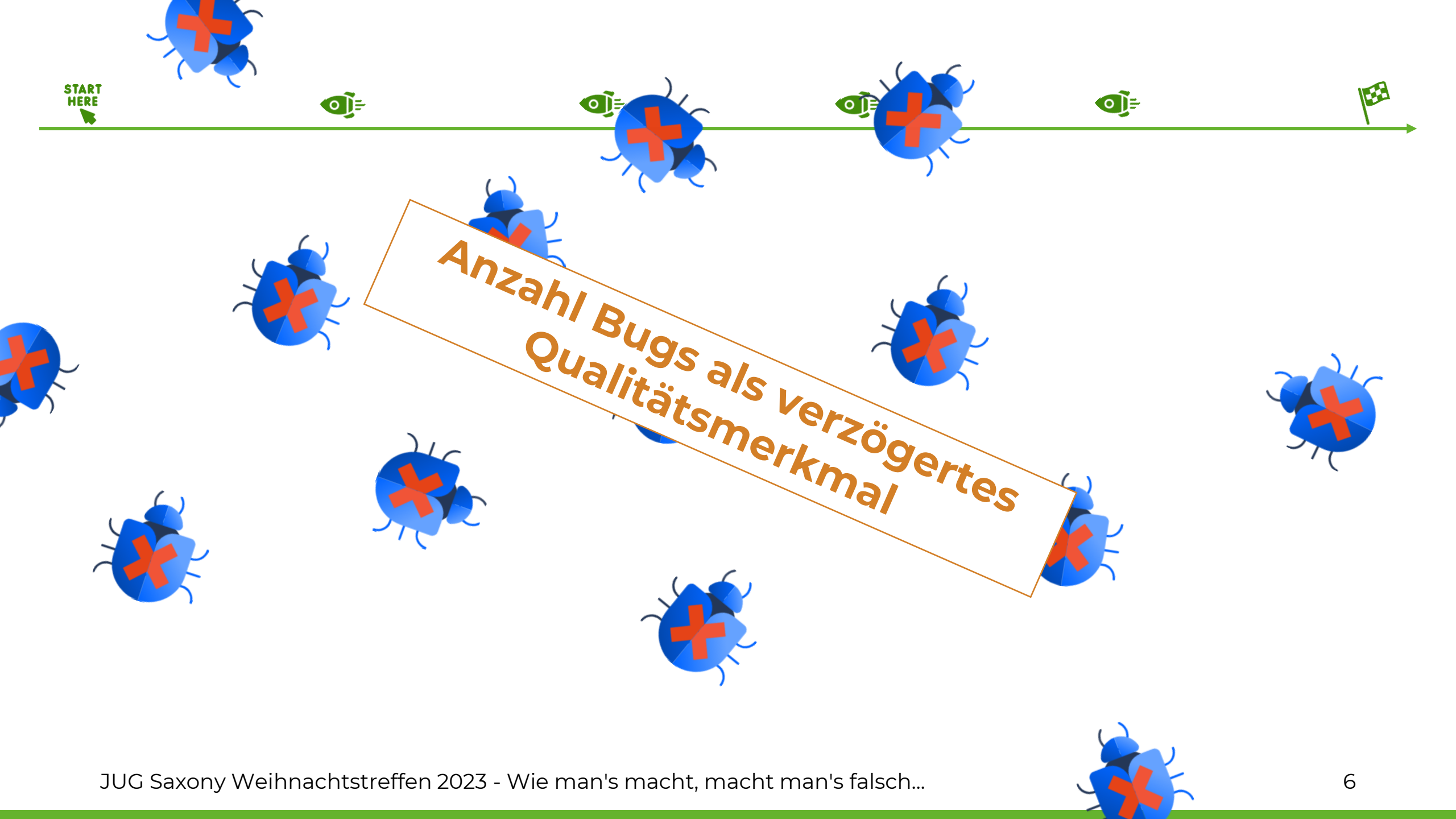
- Dresdner IT-Consulting-Unternehmen
  - Zu Hause in der Java-Welt verbunden mit neuesten Technologien und Altbewährtem (z. B. React, Angular, Python, Neo4j, etc.)
  - Langjährige Forschungskooperationen mit Hochschulen und Engagement in Communities (JUG Saxony)
- Unsere Schwerpunkte
  - Strategische, zielgerichtete und nachhaltige Entwicklung von Geschäftsanwendungen
  - Software-Qualitätsanalysen und -Sicherung
  - Köpfe hinter jQAssistant: von Entwicklung über Workshops bis hin zu Consulting
- Unsere Kunden
  - Kleine Auswahl: ITZBund, Sächsische Aufbaubank-Förderbank, ASML, GlobalFoundries, Thyssenkrupp Steel, Telekom, COOP Schweiz

Hot Take:

Softwareentwicklung ist total einfach

- Unser Kunde sagt uns in klaren Worten, was er braucht
- Wir analysieren die Anforderungen und deren Impact auf das System und bestimmen präzise den Aufwand
- Wir planen die Änderungen zeitnah ein und setzen diese konform zu unseren Qualitätsansprüchen um
- Unsere automatisierte Pipeline qualifiziert die Änderungen und ein/e weitere/r Entwickler/in führt nicht-automatisierbare Checks durch
- Nach erfolgreicher Prüfung wird planmäßig ein neuer Release gebaut und erfolgreich ausgerollt
- Unser Kunde ist glücklich, da alles ohne Probleme lief

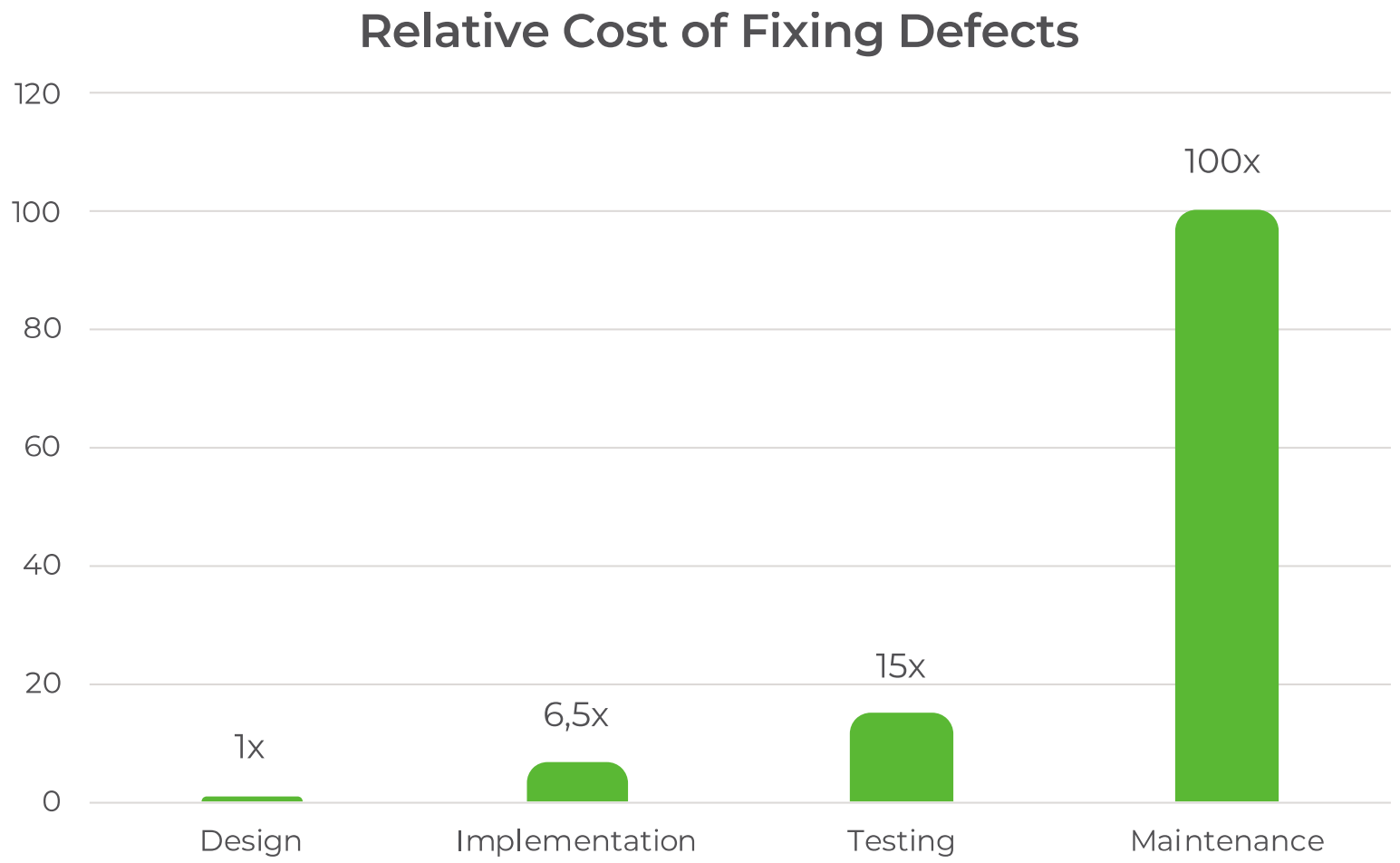

HOLD UP
WAIT A SECOND
makeameme.org


START HERE
Anzahl Bugs als verzögertes Qualitätsmerkmal

Merke:

Änderungen ziehen meist weitere Änderungen nach sich.

START HERE



Quelle: IBM Systems Sciences Institute

START HERE

The Register®

This article is more than **1 year old**

Everyone cites that 'bugs are 100x more expensive to fix in production' research, but the study might not even exist

It's probably still true, though, says formal methods expert

Tim Anderson

Thu 22 Jul 2021 // 10:01 UTC

- ~~Unser Kunde sagt uns in klaren Worten, was er braucht~~
- Wir analysieren die Anforderungen und deren Impact auf das System und bestimmen ~~präzise~~ den Aufwand
- Wir planen die Änderungen ~~zeitnah~~ ein und setzen diese ~~konform~~ zu unseren „Qualitätsansprüchen“ um
- Unsere automatisierte Pipeline qualifiziert die Änderungen und ein/e weitere/r Entwickler/in führt nicht-automatisierbare Checks durch
- Nach ~~erfolgreicher~~ Prüfung wird ~~planmäßig~~ ein neuer Release gebaut und ~~erfolgreich~~ ausgerollt
- Unser Kunde ~~ist glücklich, da alles ohne Probleme lief~~ stellt Bug Tickets ein

Um das klar zu stellen:
Softwareentwicklung ist komplex

Es kann immer etwas schief gehen

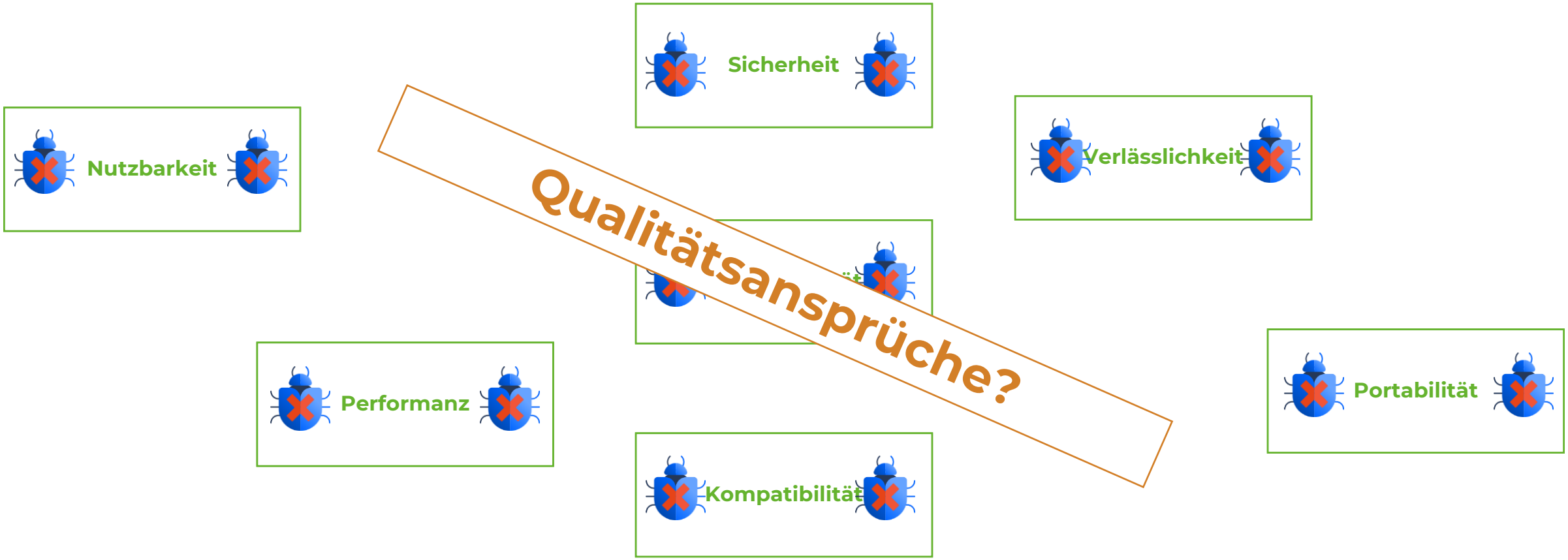
Dass mehr als 1 Person involviert ist, macht es nicht einfacher

Aber auch:

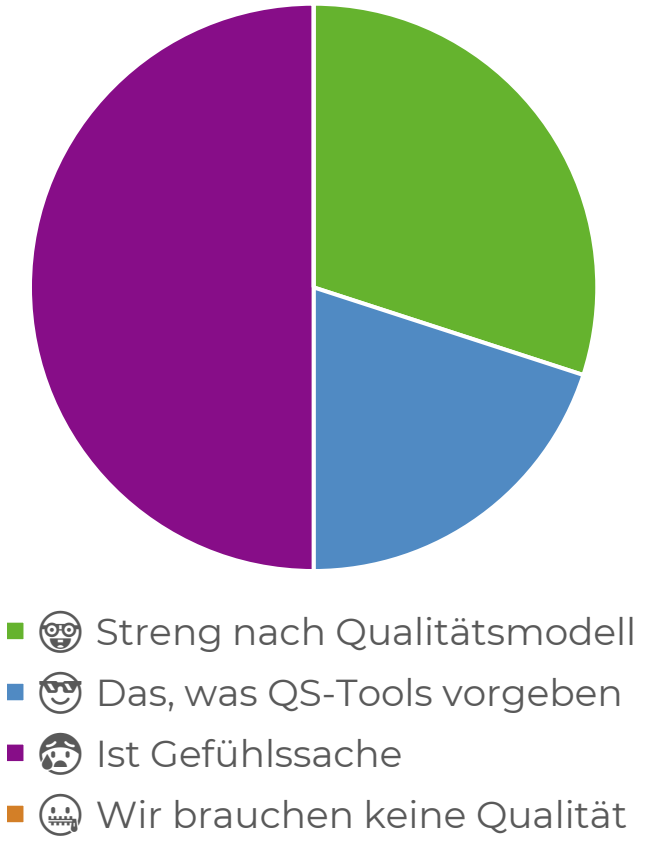
Je eher wir gegensteuern, desto geringer sind die Kosten

Die Frage ist:

Wie können wir gegensteuern?


Sicherheit
Nutzbarkeit
Qualitätsansprüche?
Verlässlichkeit
Performanz
Portabilität
Kompatibilität

Wie bestimmt Ihr, ob Eure Software gut ist?

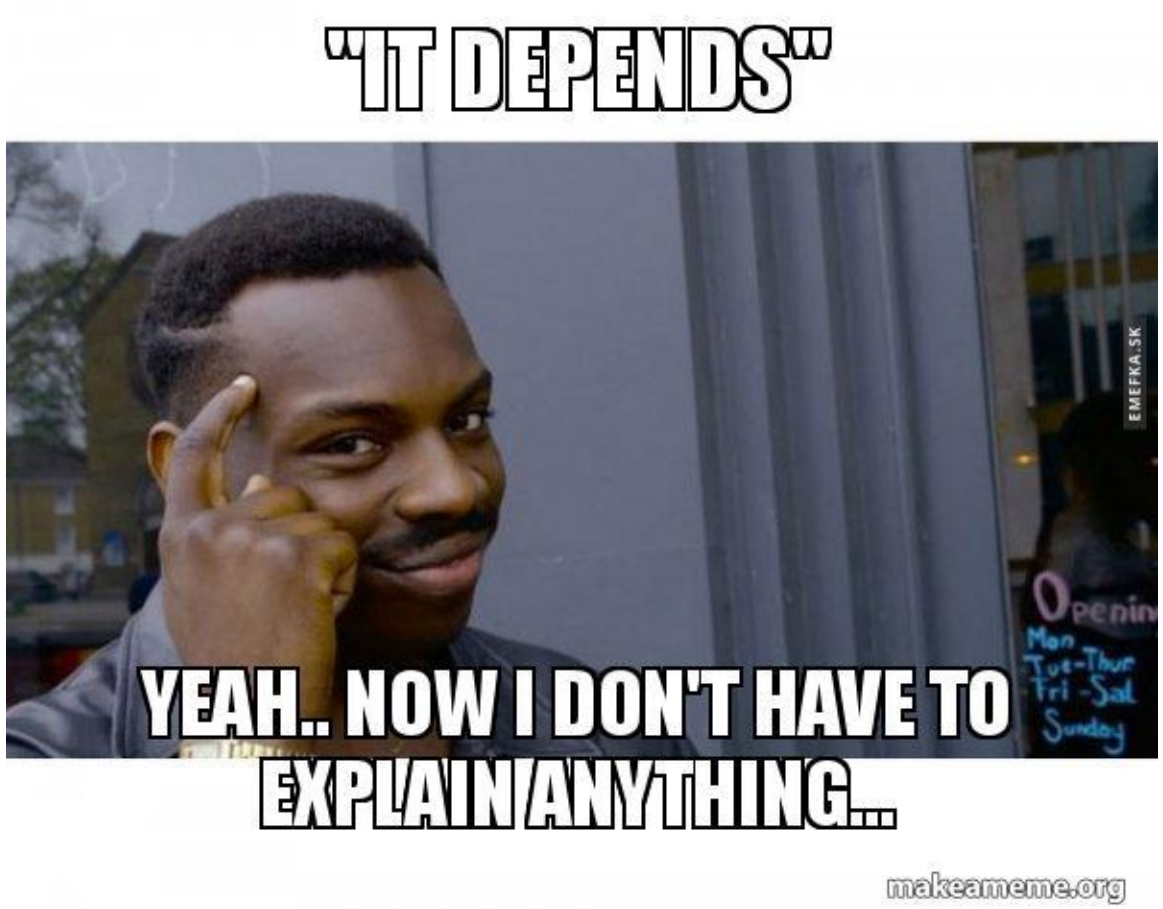


Unter **Softwarequalität** versteht man die Gesamtheit der **Merkmale** und **Merkmalswerte** eines Softwareprodukts, die sich auf dessen **Eignung** beziehen, festgelegte oder vorausgesetzte **Erfordernisse** zu erfüllen.

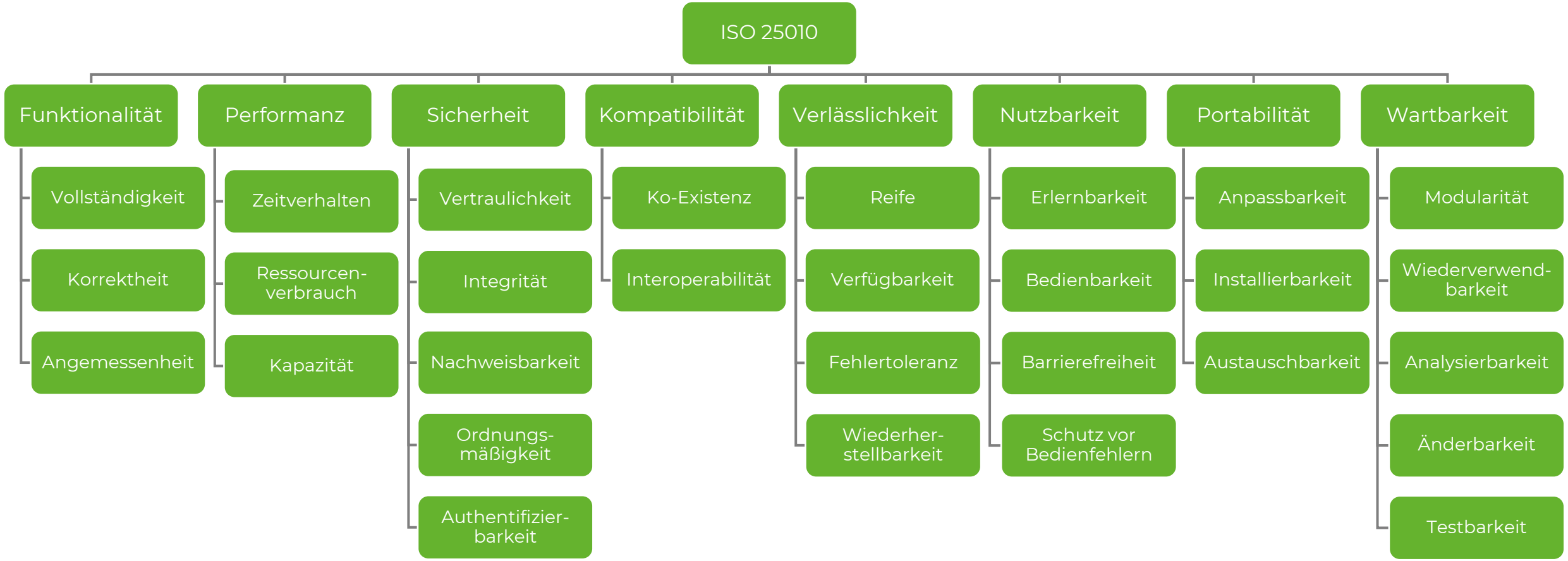
- Helmut Balzert -

- Jede Software ist individuell, daher muss auch das Qualitätsmodell individuell erstellt werden
  - Welche Vorgaben und (Geschäfts-)Ziele definiert das Unternehmen?
  - Welche Ansprüche haben die Nutzer bzw. Kunden?
  - Welche technischen Rahmenbedingungen existieren?
  - Was sind die Bedürfnisse der Softwareentwickler, QA/QS-Abteilung, etc.?

- Resultierendes Qualitätsmodell ist zumeist ein Trade-Off

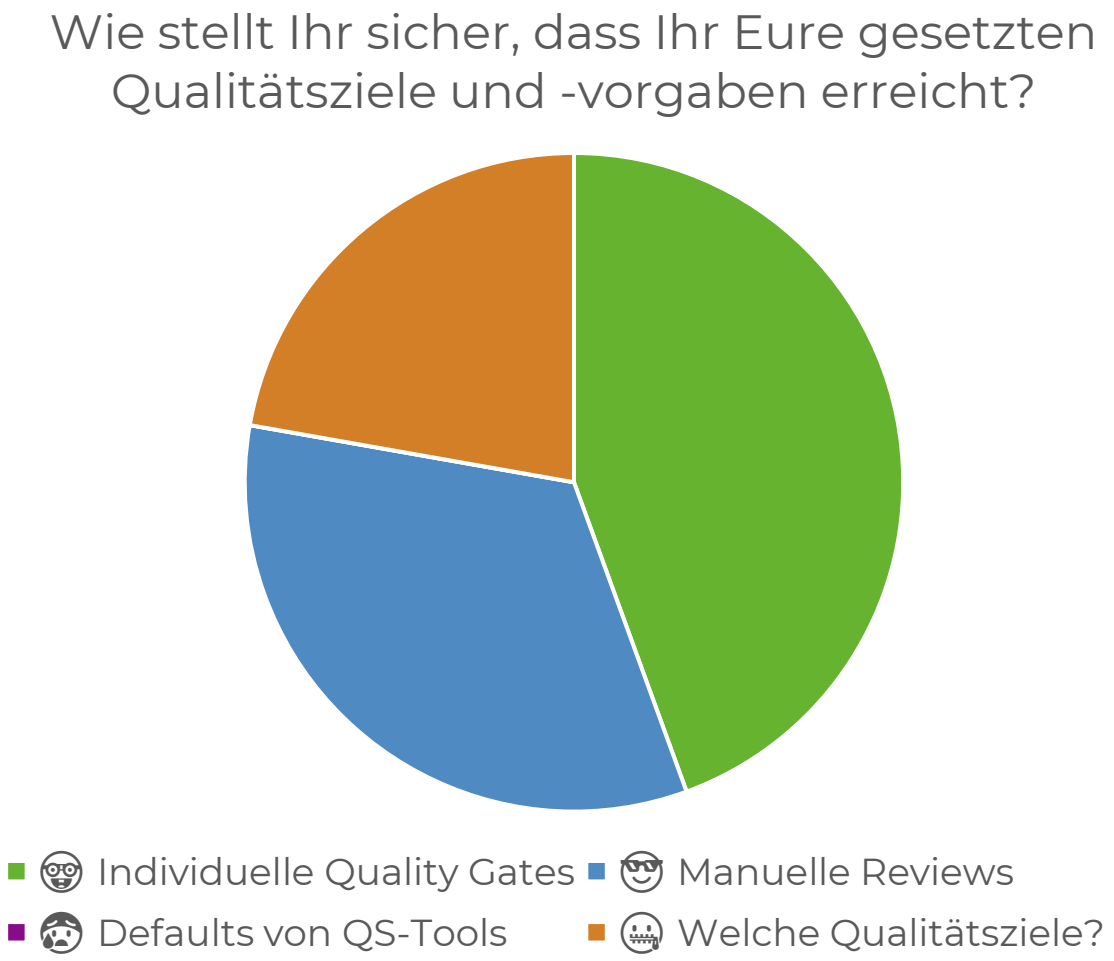


https://makeameme.org/meme/it-depends-yeah


ISO 25010
Funktionalität
Vollständigkeit
Korrektheit
Angemessenheit
Performanz
Zeitverhalten
Ressourcen-verbrauch
Kapazität
Sicherheit
Vertraulichkeit
Integrität
Nachweisbarkeit
Ordnungs-mäßigkeit
Authentifizier-barkeit
Kompatibilität
Ko-Existenz
Interoperabilität
Verlässlichkeit
Reife
Verfügbarkeit
Fehlertoleranz
Wiederher-stellbarkeit
Nutzbarkeit
Erlernbarkeit
Bedienbarkeit
Barrierefreiheit
Schutz vor Bedienfehlern
Portabilität
Anpassbarkeit
Installierbarkeit
Austauschbarkeit
Wartbarkeit
Modularität
Wiederverwend-barkeit
Analysierbarkeit
Änderbarkeit
Testbarkeit

Du kennst jetzt die zu optimierenden Qualitätsmerkmale?

Woher weißt du, ob du die gesetzten Ziele erreichst?

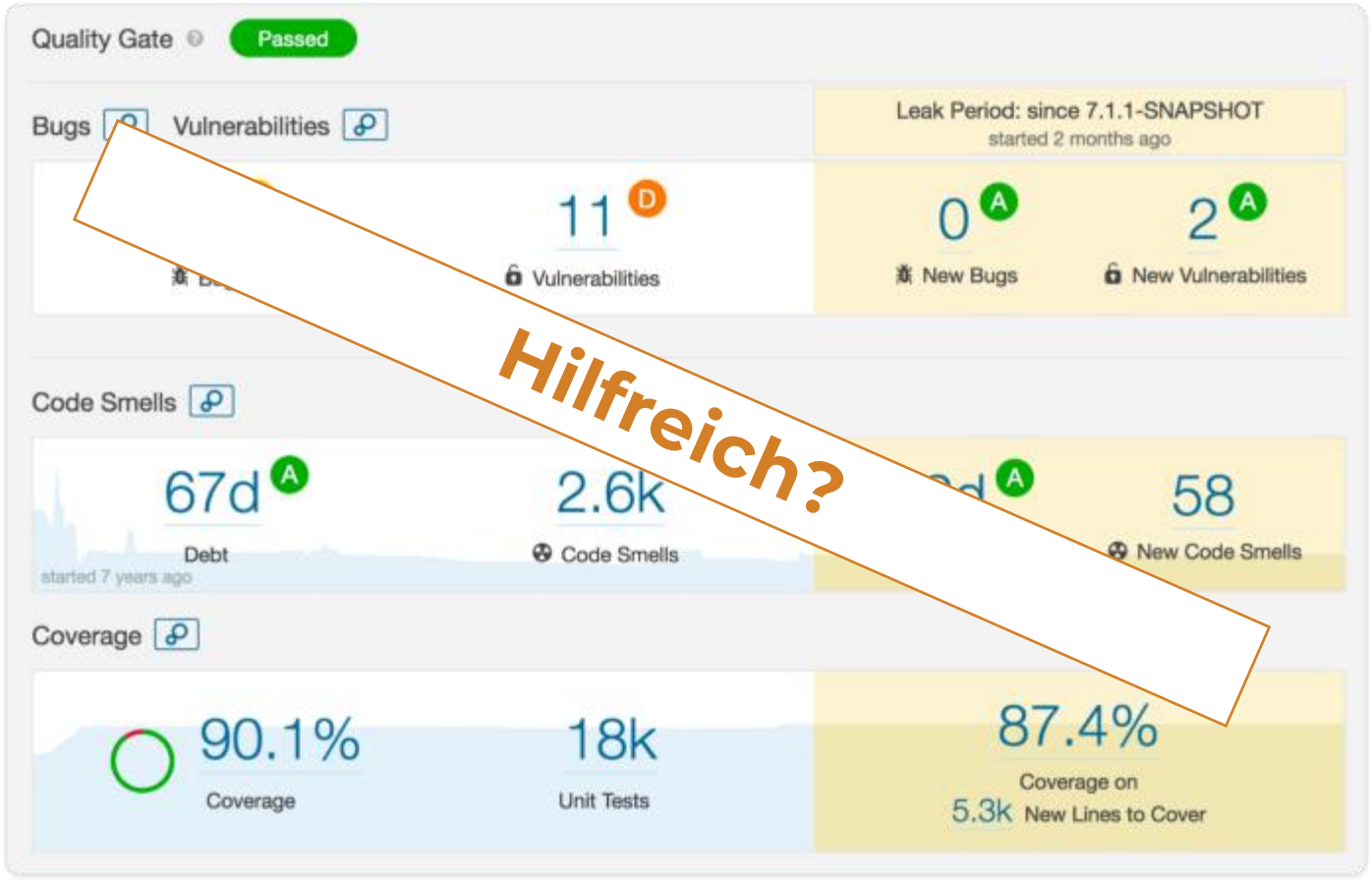

Wie stellt Ihr sicher, dass Ihr Eure gesetzten Qualitätsziele und -vorgaben erreicht?
🤓 Individuelle Quality Gates
😎 Manuelle Reviews
😰 Defaults von QS-Tools
🤐 Welche Qualitätsziele?

Softwarequalität muss messbar werden!

Eine Softwarequalitätsmetrik ist eine Funktion, die eine Software-Einheit in einen **Zahlenwert** abbildet, welcher als **Erfüllungsgrad** einer **Qualitätseigenschaft** der Software-Einheit interpretierbar ist.

- IEEE Standard 1061, 1998 -

- Metriken machen Qualitätsmerkmale messbar, z.B.
  - Zeitverhalten (Performanz) -> Response Time messen
  - Modularität (Wartbarkeit) -> Kohäsion und Kopplung messen
- ABER:
  - Eine Metrik bzw. deren Messung ist die Abbildung des Systems auf eine Quantität
  - Die Quantität stellt definitionsbedingt immer nur eine Annäherung einer Qualität dar
  - Jede Qualität ist in zahlreichen Quantitäten ausdrückbar
  - Selektion der Metriken ist maßgeblich für die Qualität der Bewertung



https://medium.com/red6-es/go-for-sonarqube-ffff5b74f33a

- Metriken sind hilfreich, wenn bekannt ist,
  - warum diese erhoben werden
  - für wen die Metriken erhoben werden
  - welche Auswirkungen diese in der Praxis haben
  - wie sich diese berechnen
  - welche Werte erreicht werden müssen
  - wie die Werte verbessert werden können

- Wir lernen: Auch eine Metrik wie die Lines of Code kann hilfreich sein

Wie bestimmt man denn hilfreiche Metriken?

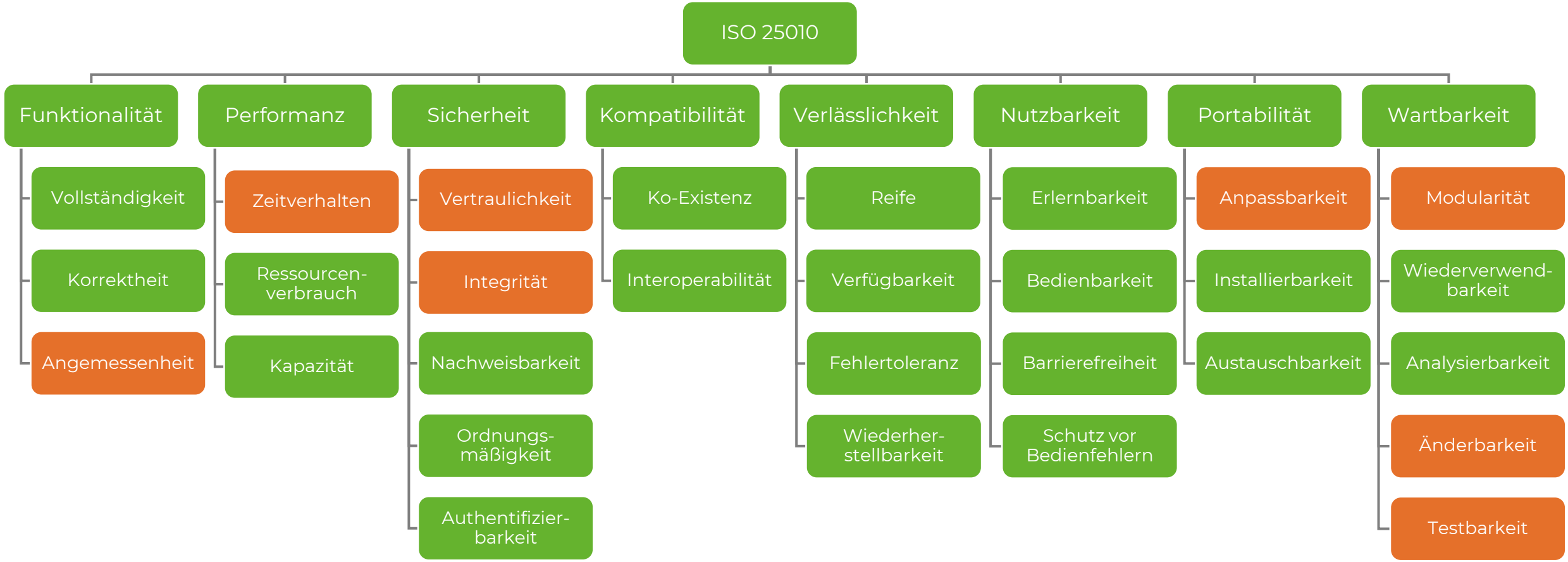
Vorgehen zur Selektion von Metriken

1. Qualitätsmodell mit Qualitäts(teil)merkmalen festlegen
2. Priorisierung und Risikobewertung der Merkmale
3. Identifikation möglicher Metriken (je Merkmal)
4. Erprobung der Metriken und Identifikation sinnvoller Grenzwerte
5. Festlegung auf initialen Satz an Metriken

- Kontinuierliche Evaluierung, Monitoring, Anpassung

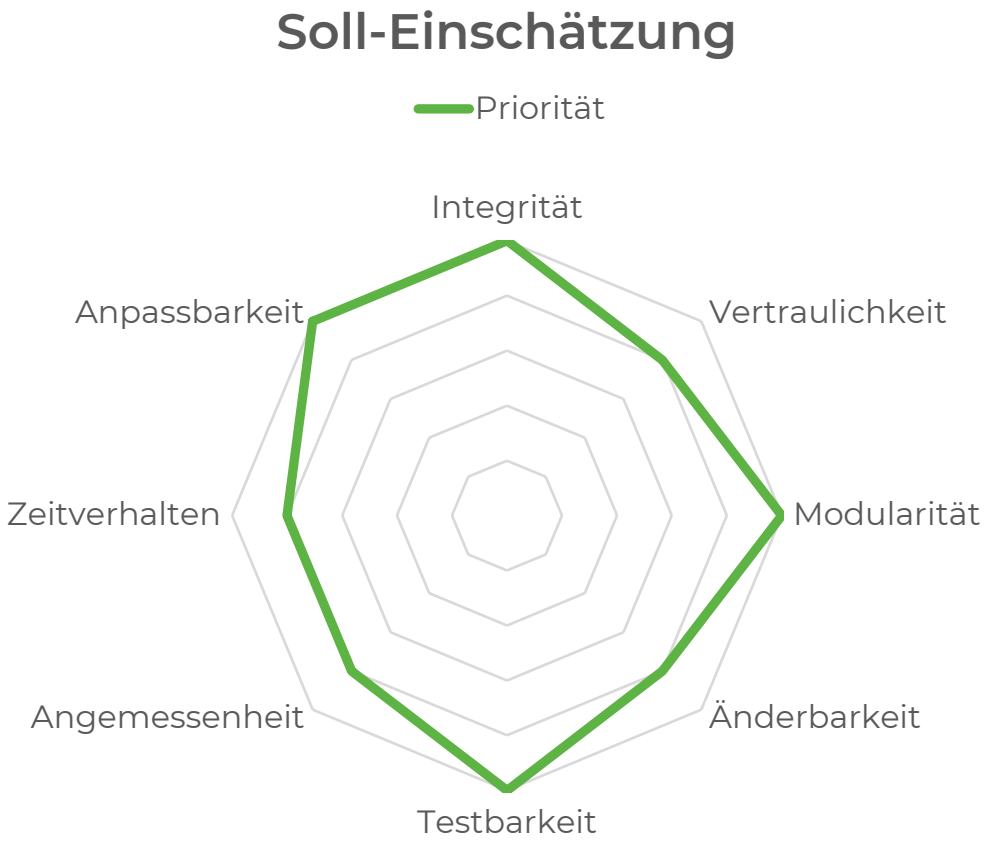
Schritt 1:

Qualitätsmodell mit Qualitäts(teil)merkmalen festlegen

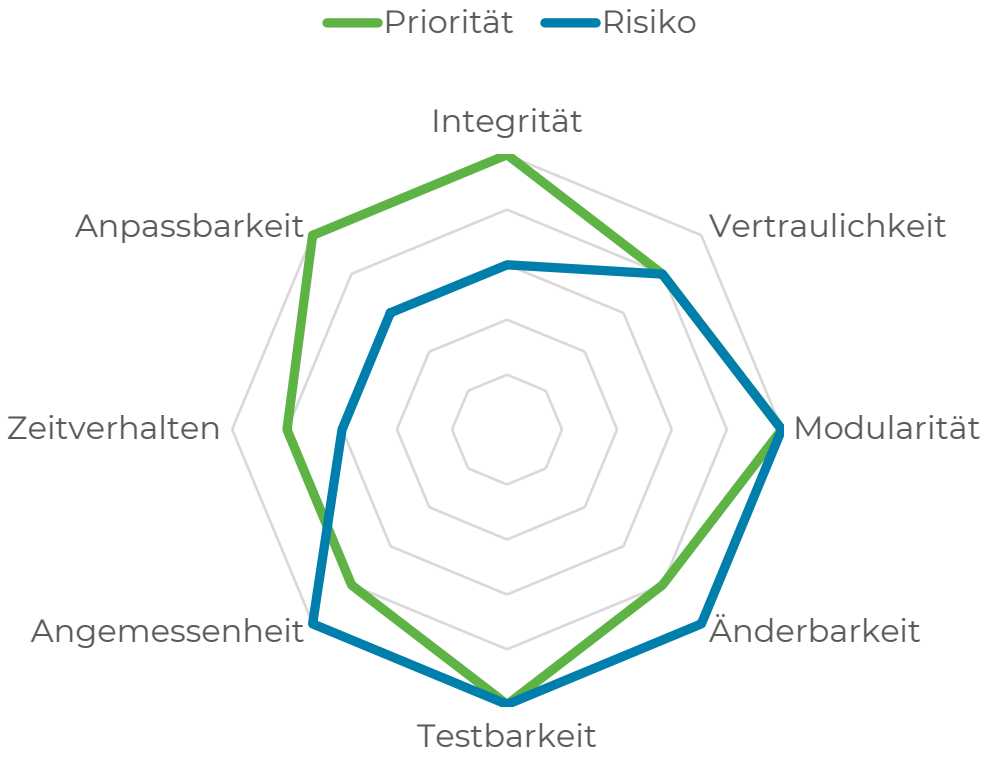

ISO 25010
Funktionalität
Vollständigkeit
Korrektheit
Angemessenheit
Performanz
Zeitverhalten
Ressourcen-verbrauch
Kapazität
Sicherheit
Vertraulichkeit
Integrität
Nachweisbarkeit
Ordnungs-mäßigkeit
Authentifizier-barkeit
Kompatibilität
Ko-Existenz
Interoperabilität
Verlässlichkeit
Reife
Verfügbarkeit
Fehlertoleranz
Wiederher-stellbarkeit
Nutzbarkeit
Erlernbarkeit
Bedienbarkeit
Barrierefreiheit
Schutz vor Bedienfehlern
Portabilität
Anpassbarkeit
Installierbarkeit
Austauschbarkeit
Wartbarkeit
Modularität
Wiederverwend-barkeit
Analysierbarkeit
Änderbarkeit
Testbarkeit

Schritt 2:

Priorisierung und Risikobewertung der Merkmale


Soll-Einschätzung
Priorität
Integrität
Vertraulichkeit
Modularität
Änderbarkeit
Testbarkeit
Angemessenheit
Zeitverhalten
Anpassbarkeit

Soll- und Risiko-Einschätzung



Schritt 3:

Identifikation möglicher Metriken (je Merkmal)

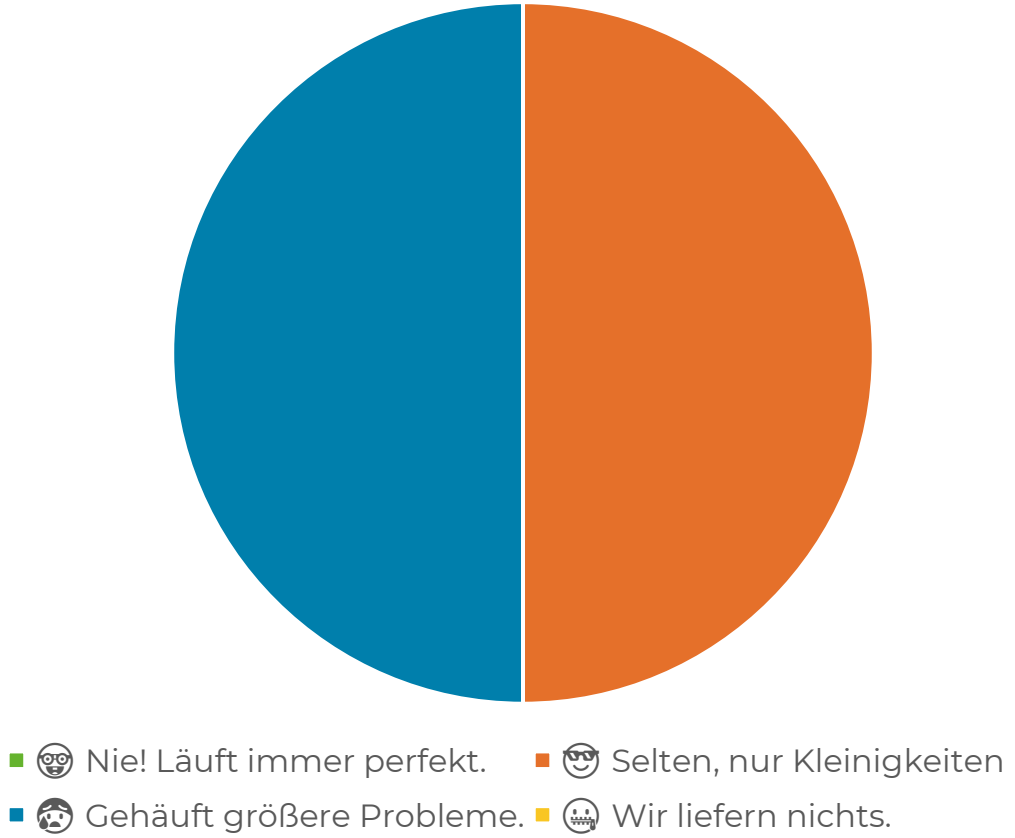
- Beispiel „Änderbarkeit“ (Wartbarkeit)
  - Metrik 1: zyklomatische Komplexität
    - z. B. Summe der zyklomatischen Komplexitäten aller Klassen
  - Metrik 2: Kopplungsgrad zwischen Modulen
    - z. B. Durchschnitt der Instabilitätsmaße aller Klassen (FanIn / (FanIn + FanOut))
  - Metrik 3: Kohäsionsgrad innerhalb eines Moduls
    - Geschlossene Lösung einer logischen Aufgabe innerhalb einer Klasse
    - z. B. Durchschnitt der LCOM-Werte aller Klassen (Lack of Cohesion of Methods)

Schritt 4:

Erprobung der Metriken und Identifikation des Einflusses auf nachgelagerte Änderungen

START HERE

Wie häufig müsst ihr an ausgelieferten Änderungen Nacharbeiten durchführen?



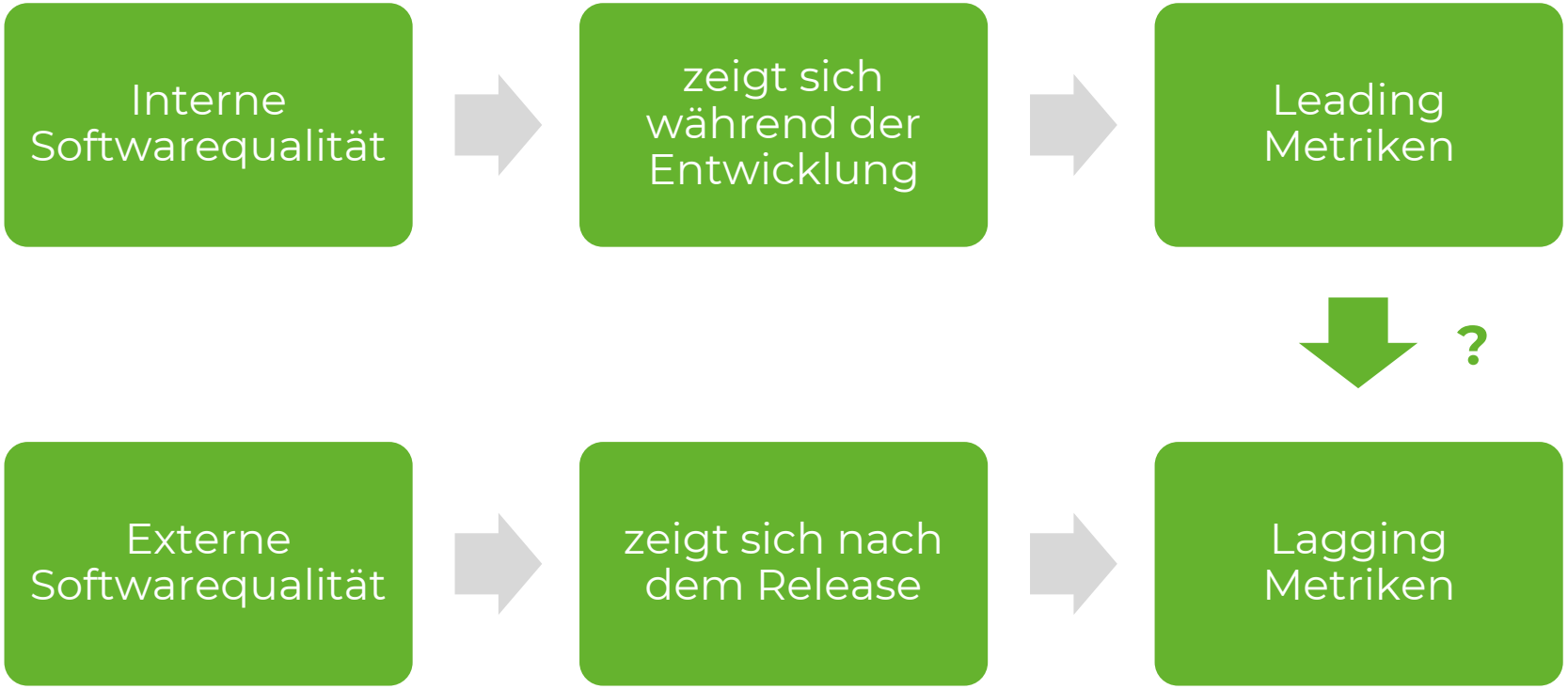
Merke: Änderungen ziehen Änderungen nach sich.

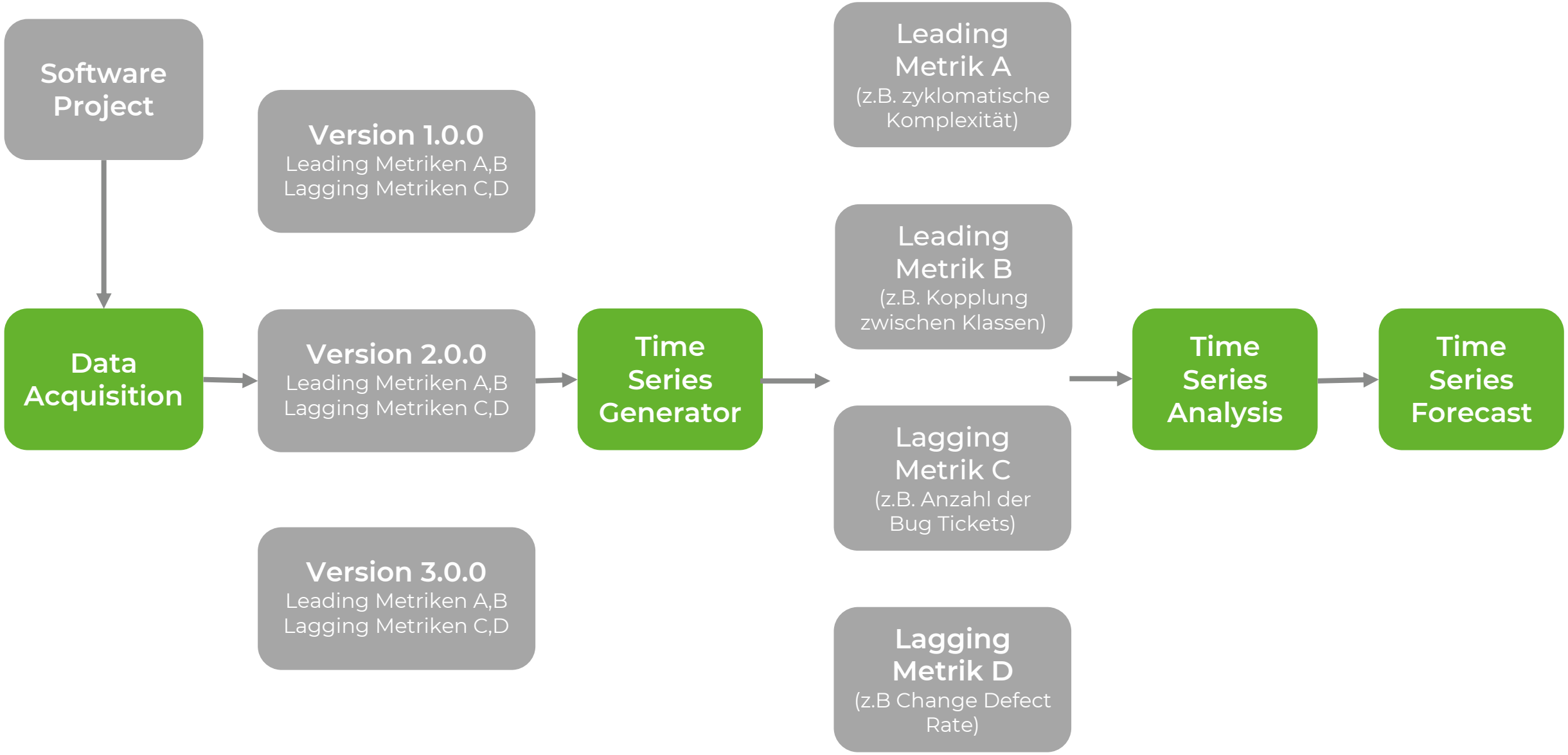
Warum?

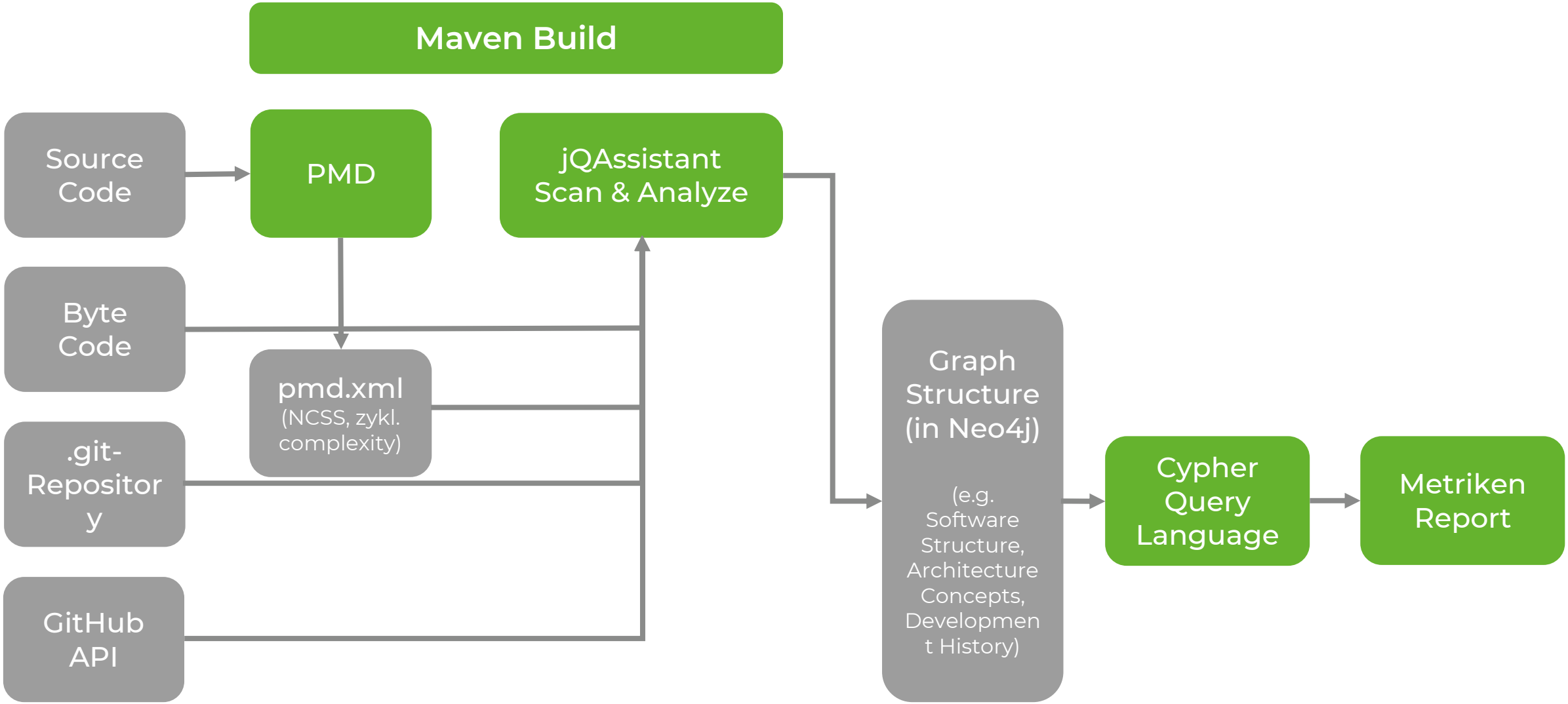
Die initialen Änderungen erfüllen nicht / unzureichend die Qualitätsziele.

Wie wird dies deutlich?

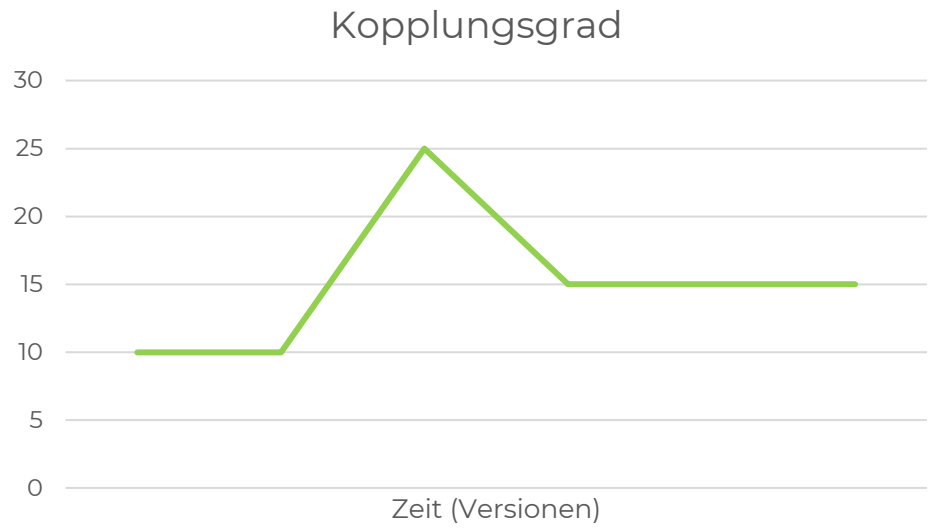
- Nichterfüllung der Qualitätsziele wirkt sich negativ aus auf z. B.
  - Entwicklungskosten und -dauer
  - Abschätzbarkeit und Verlässlichkeit der Planung
  - Laufzeiteigenschaften wie Response Time oder Ressourcenverbrauch
  - Nutzer-/Kundenzufriedenheit
  - Risiko ist im Kontext des Projekts abhängig von den jeweils betrachteten Zielen
- **Lagging Metriken**


Interne Softwarequalität
zeigt sich während der Entwicklung
Leading Metriken
?
Externe Softwarequalität
zeigt sich nach dem Release
Lagging Metriken

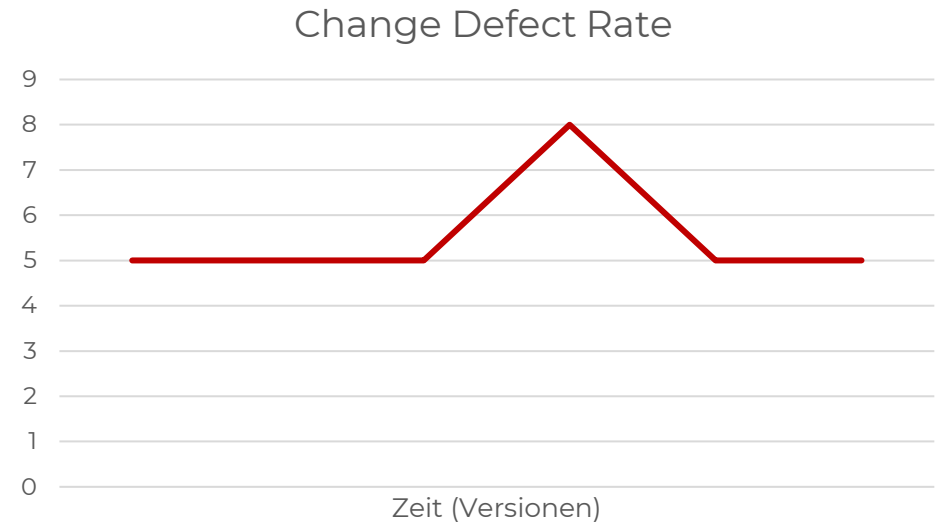

Software Project
Data Acquisition
Version 1.0.0
Leading Metriken A,B
Lagging Metriken C,D
Version 2.0.0
Leading Metriken A,B
Lagging Metriken C,D
Version 3.0.0
Leading Metriken A,B
Lagging Metriken C,D
Time Series Generator
Leading Metrik A
(z.B. zyklomatische Komplexität)
Leading Metrik B
(z.B. Kopplung zwischen Klassen)
Lagging Metrik C
(z.B. Anzahl der Bug Tickets)
Lagging Metrik D
(z.B Change Defect Rate)
Time Series Analysis
Time Series Forecast

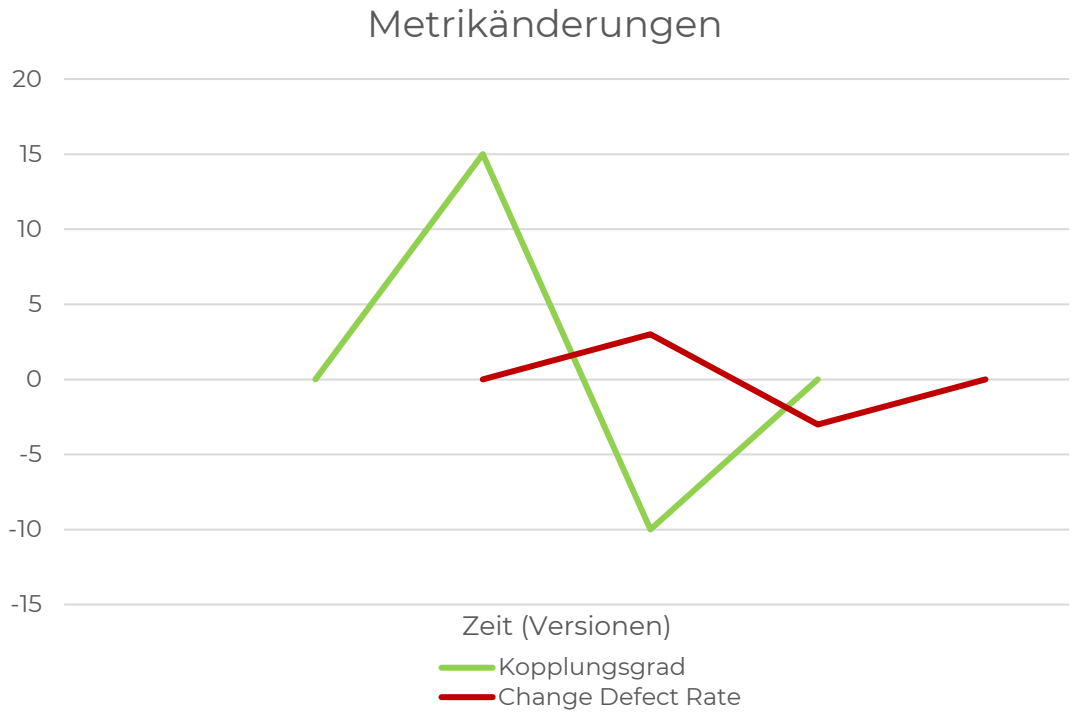

Maven Build
Source Code
PMD
jQAssistant Scan & Analyze
Byte Code
pmd.xml
(NCSS, zykl. complexity)
.git-Repository
GitHub API
Graph Structure (in Neo4j)
(e.g. Software Structure, Architecture Concepts, Development History)
Cypher Query Language
Metriken Report

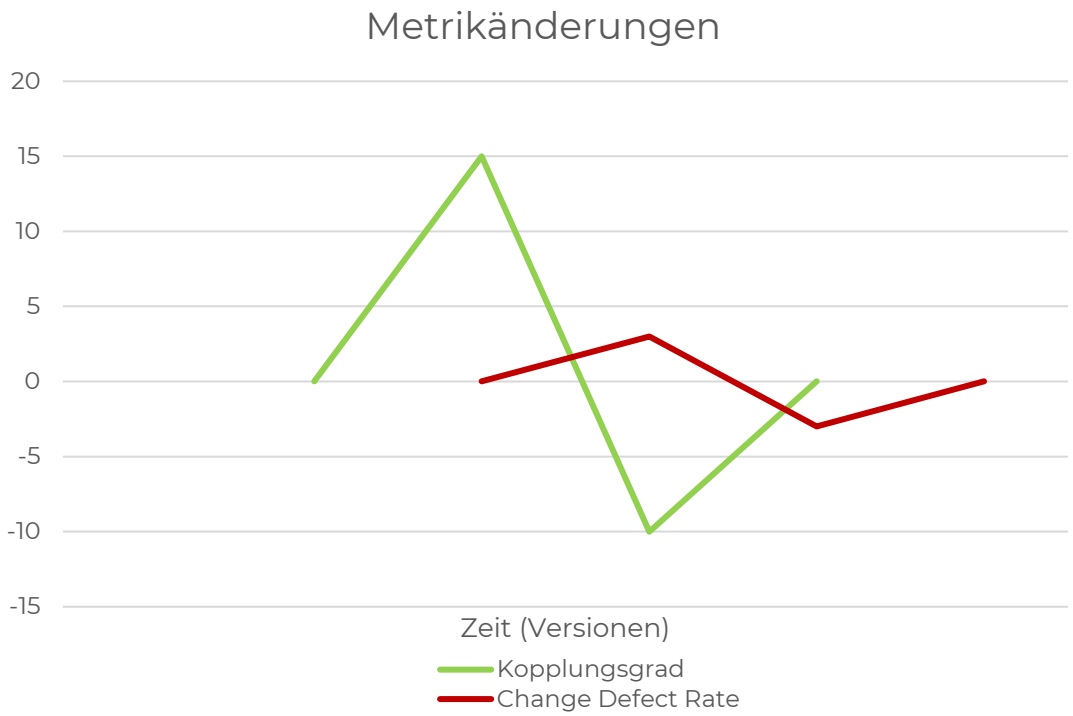
Leading Metrik B
(z.B. Kopplung zwischen Klassen)



Lagging Metrik C
(z.B. Change Defect Rate)



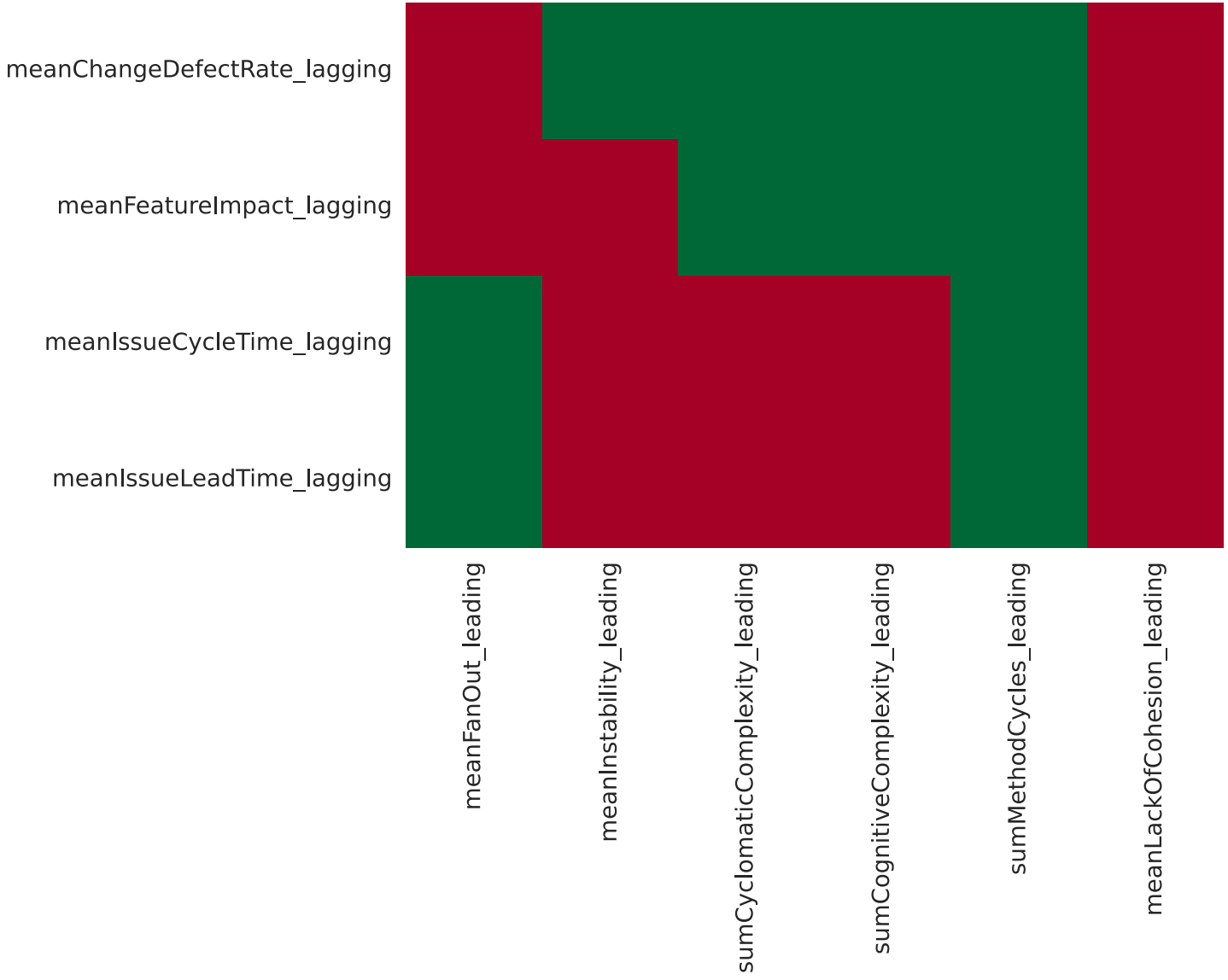




Hypothesentest (Granger Causality Test)

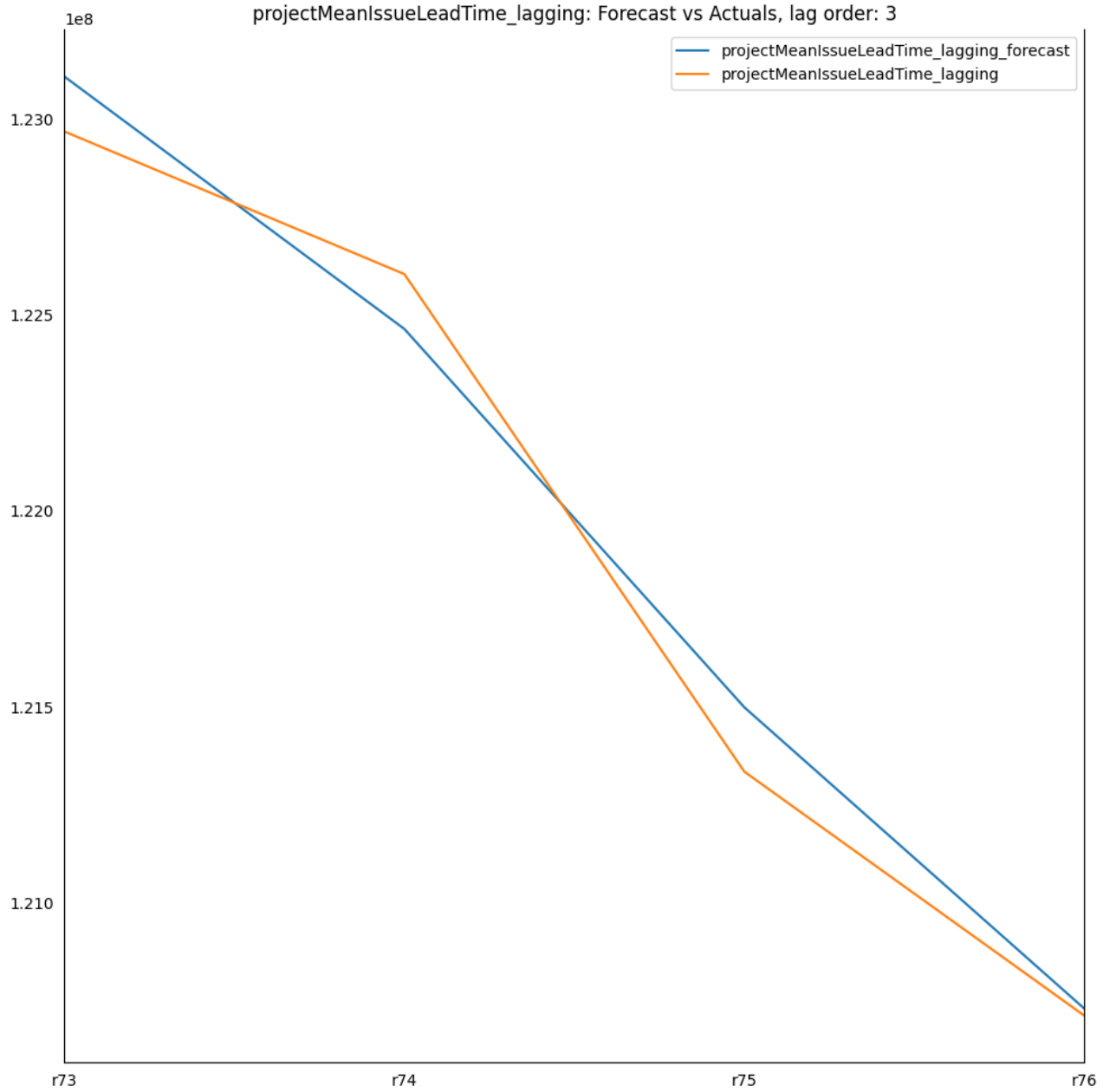
Forecast Model (Vector Autoregression)

- Beispielprojekt (277 kLOC)
- Analyse auf Projektebene
- 21 inkludierte Versionen
- Fokus auf Wartbarkeit



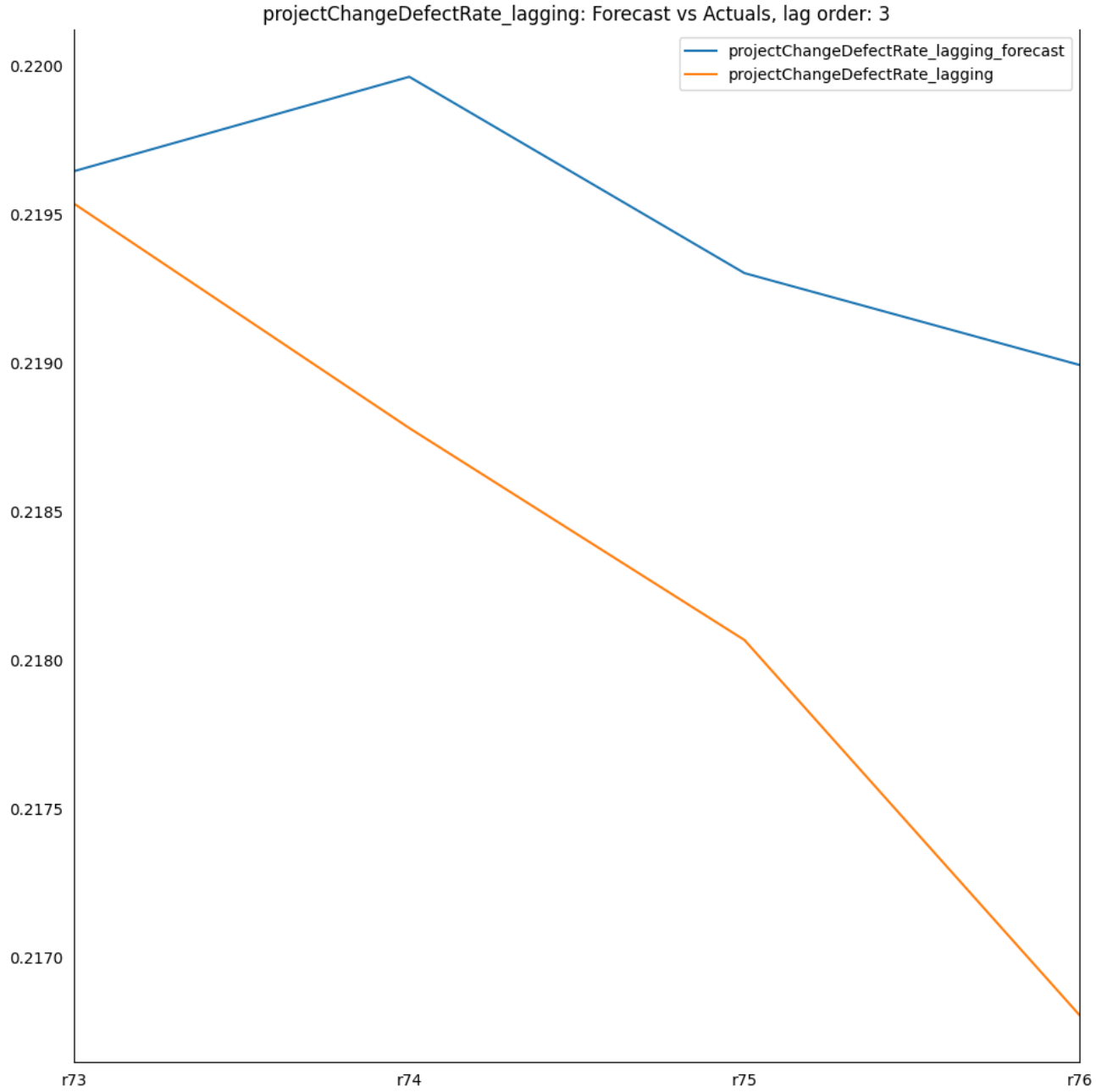
- Beispielprojekt (277 kLOC)
- Analyse auf Projektebene
- 21 inkludierte Versionen

- Forecast für Mean Issue Lead Time



- Beispielprojekt (277 kLOC)
- Analyse auf Projektebene
- 21 inkludierte Versionen

- Forecast für Change Defect Rate



Änderungen ziehen Änderungen nach sich.

Warum?

Die initialen Änderungen erfüllen nicht / unzureichend die Qualitätsziele.

Wie wird dies deutlich?

Durch ausschlagende Lagging-Metriken.

Was kann man tun?

Bewusstes Optimieren von Leading-Metriken, die diese Lagging-Metriken beeinflussen.

Schritt 5:

Festlegung auf initialen Satz an Metriken

- Beispiel „Änderbarkeit“ (Wartbarkeit)
  - Metrik 1: zyklomatische Komplexität
    - z.B. Summe der zyklomatischen Komplexitäten aller Klassen
  - Metrik 2: Kopplungsgrad zwischen Modulen
    - z.B. Durchschnitt der Instabilitätsmaße aller Klassen (FanIn / (FanIn + FanOut))
  - ~~Metrik 3: Kohäsionsgrad innerhalb eines Moduls~~
    - ~~Geschlossene Lösung einer logischen Aufgabe innerhalb einer Klasse~~
    - ~~z.B. Durchschnitt der LCOM-Werte aller Klassen (Lack of Cohesion of Methods)~~

Okay. Und was bedeutet das jetzt für mich?

- Qualität ist die Fähigkeit, Kundenanforderungen erfüllen zu können
- Prüft, welche Qualitätsziele für euch und eure Kunden relevant sind
- Wählt Metriken, die zu euren Qualitätszielen passen (Leading)
- Hört auf, blind auf Standardregelsätze von QS-Tools zu optimieren
- Fangt an, Auswirkungen eurer Optimierungen zu monitoren (Lagging)
- Habt den Mut, eure Metriken regelmäßig zu überarbeiten

VIELEN DANK

Noch Fragen?

BUSCHMAIS

KONTAKT

Stephan Pirnbaum

stephan.pirnbaum@buschmais.com

+49 351 320923-22

Social Media

@spirnbaum

stephan-pirnbaum

BUSCHMAIS GbR

Leipziger Straße 93

01127 Dresden

Tel. +49 351 3209230

info@buschmais.com

www.buschmais.de