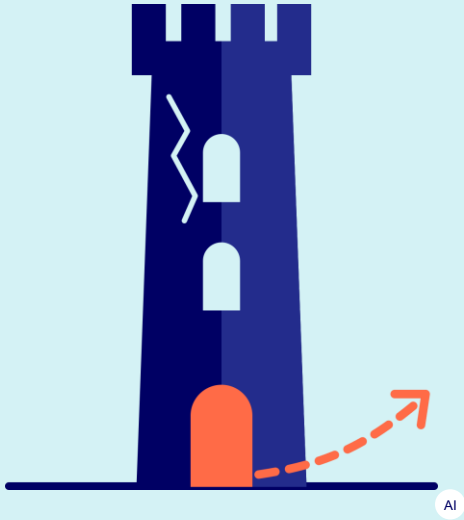
parcIT

Mitglied der
ATRUVIA GRUPPE

Teamscale
by CQSE



Ausbruch aus dem Elfenbeinturm

Nach allen Regeln der Codeanalyse

Maik Heene, Johannes Veihelmann – München – 04.11.2026

Bevor wir anfangen



Johannes Veihelmann

Teamscale

Technischer Berater für Software Qualität

Begleitet Teams bei der Einführung statischer Codeanalyse

&



Maik Heene

parcIT GmbH

Testautomatisierungsmanager im Banken-Umfeld

Zweifacher Papa - Regeldiskussionen bin ich gewohnt

Dienstagmorgen. Der Build bricht.

Wer hat das entschieden?



Wir ziehen die Kommentare nach

Zwanzig Minuten

Niemand liest sie je wieder

Wer hat diese Regel eigentlich aktiviert?



In der Retrospektive stellt man genau diese Frage

Niemand im Raum kann sie beantworten

Teamscale by CQSE parcIT

Meine Erfahrung: Zwei Wege, wie ein Regelset entsteht

Wie Regelsets wirklich entstehen

Der Erbfall

- Du kommst in eine Firma, das Werkzeug ist längst da.
- Die Regeln sind geerbt. Die Begründung nicht.

Der Neustart

- Jemand entdeckt das Werkzeug frisch für sich.
- Einschalten, Defaults übernehmen, Finger in den Wind: „**Wird schon passen**.“

Bittere Erfahrung: Ein Regelset, für das niemand geradesteht

Beim Testen haben wir das längst gelernt

Wie Regelsets wirklich entstehen

„Beim Testen akzeptiert niemand mehr, dass die Testpyramide für jede Architektur passt.

Bei Coding-Rules akzeptieren wir genau das seit Jahren!“

Der ISTQB definiert Teststufen

aber nicht deren Ausprägung

Die Ausprägung folgt

aus Architektur und Risiko

daher aus dem Kontext

und der ist bei jedem anders

Warum brauchen wir statische Codeanalyse?



Menschen machen Fehler

„KI schreibt doch fehlerfrei.“



Code wird öfter gelesen

„Lesen tut jetzt die KI.“



Architektur erodiert

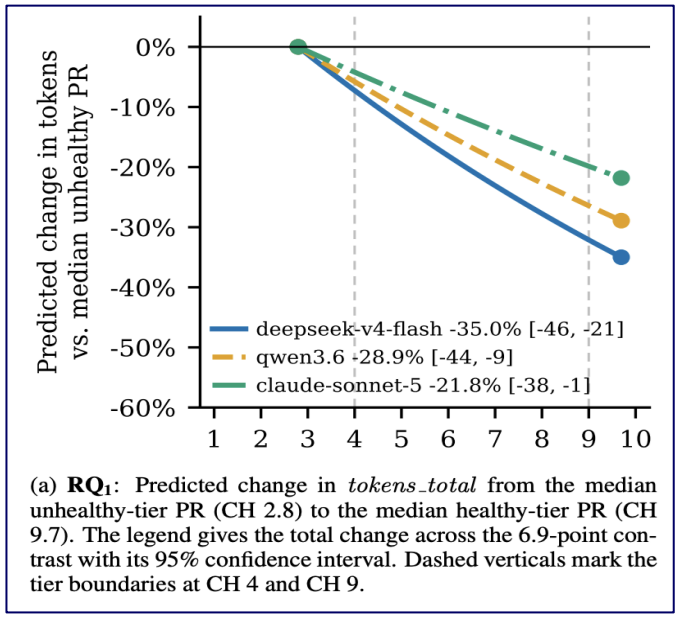
„KI kann das auf Knopfdruck.“



Reviews skalieren nicht

„KI reviewt in Sekunden.“

Schlechte Software-Qualität ist auch für Coding Agents ein Problem



(a) **RQ$_1$**: Predicted change in *tokens_total* from the median unhealthy-tier PR (CH 2.8) to the median healthy-tier PR (CH 9.7). The legend gives the total change across the 6.9-point contrast with its 95% confidence interval. Dashed verticals mark the tier boundaries at CH 4 and CH 9.

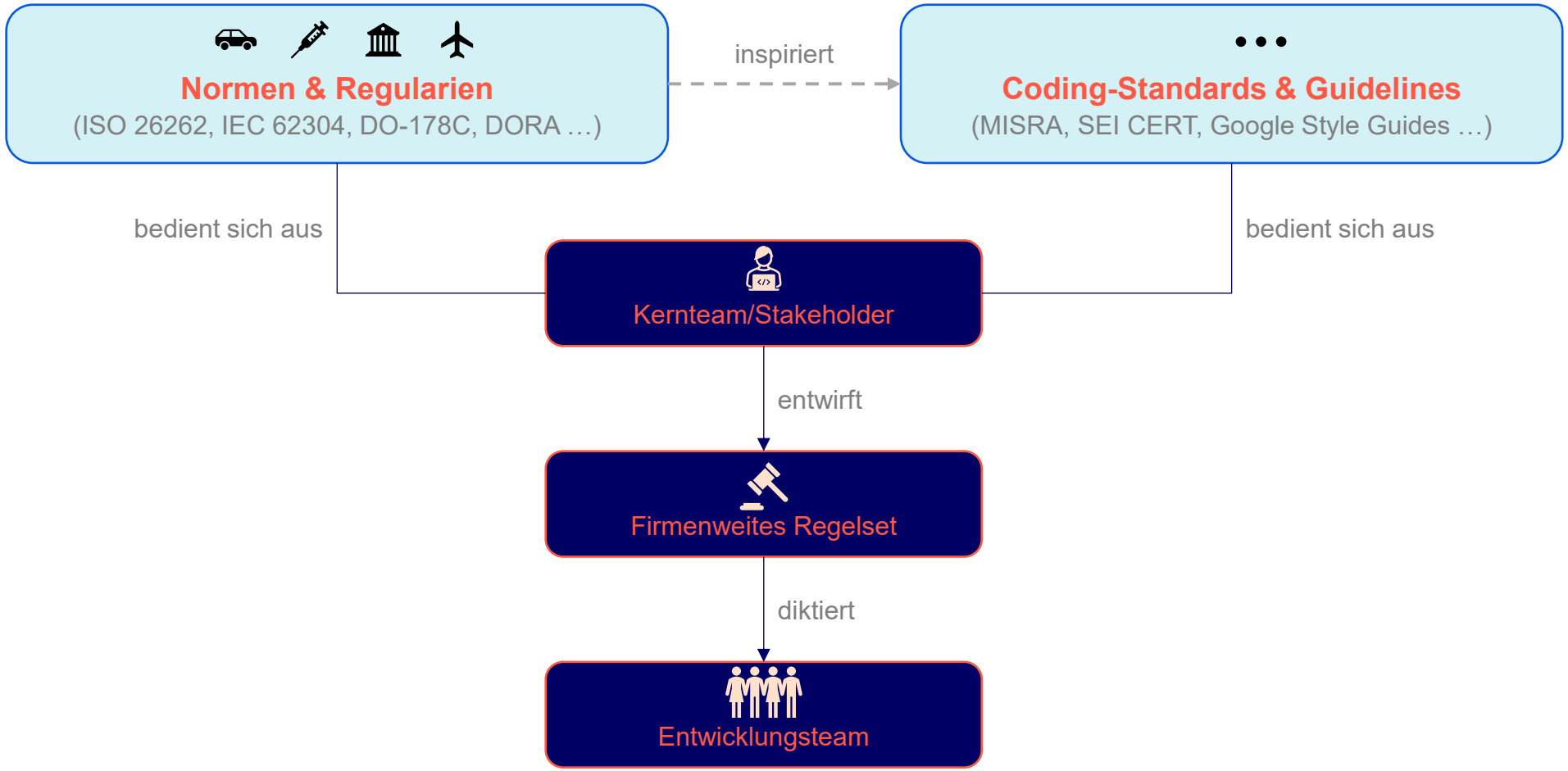
Studie: Oktober 2026 (arXiv) · Messung Aug. 2026 mit DeepSeek V4 Flash, Qwen3.6 und Claude Sonnet 5

Schlechter Code kostet auch Agenten mehr

- Gesunder statt ungesunder Code: **22–35 % weniger Tokens** bei allen 3 getesteten Modellen
- KI generiert Code schneller und die Wartungsschulden wachsen mit

Abb.: L. Gullstrand Heander, M. Borg, E. Söderberg, C. Rydenfält – „CodeHealth Predicts the Token Cost of Agentic Code Review“, arXiv 2026, CC BY 4.0 – https://arxiv.org/abs/2610.21914

Wie entstehen Regelsets in Unternehmen?



Beispiele für eine Fehlkonfiguration…

Documentation › Comment Completeness › Interface comment completeness

Interface comment missing for `CONTENT_TYPE_XML_ADX_GZIP`

Missing (or empty) interface documentation makes it hard to get a quick overview of the interface provided by a class or API. Add appropriate documentation com

What Does This Check Look For?

This check detects interfaces which are not documented with an explanatory comment. This also includes cases where comments are present, but don't add any a
tions, public classes and their public members.

Show more

dhis-2/dhis-web-api/src/main/java/org/hisp/dhis/webapi/utils/ContextUtils.java:88

Code | Introduction | Tasks | Properties | Exclusion Events

```
78      public static final String CONTENT_TYPE_CSS = "text/css; charset=UTF-8";
79
80      public static final String CONTENT_TYPE_XML = "application/xml; charset=UTF-8";
81
82      public static final String CONTENT_TYPE_XML_ADX = "application/adx+xml; charset=UTF-8";
83
84      public static final String CONTENT_TYPE_XML_GZIP = "application/xml+gzip";
85
86      public static final String CONTENT_TYPE_XML_ZIP = "application/xml+zip";
87
88      public static final String CONTENT_TYPE_XML_ADX_GZIP = "application/adx+xml+gzip";
89
90      public static final String CONTENT_TYPE_XML_ADX_ZIP = "application/adx+xml+zip";
91
92      public static final String CONTENT_TYPE_CSV = "application/csv; charset=UTF-8";
93
94      public static final String CONTENT_TYPE_TEXT_CSV = "text/csv";
95
96      public static final String CONTENT_TYPE_CSV_GZIP = "application/csv+gzip";
97
98      public static final String CONTENT_TYPE_CSV_ZIP = "application/csv+zip";
```


Teamscale by CQSE parcIT

Beispiele für eine Fehlkonfiguration…

```
41      const fooBar = "Ich bin ein String";
```

5.2.5 Constants

Immutable: `CONSTANT_CASE` indicates that a value is *intended* to not be changed, and *may* be used for values that can technically be modified (i.e. values that are not deeply frozen) to indicate to users that they must not be modified.

```
const UNIT_SUFFIXES = {
  'milliseconds': 'ms',
  'seconds': 's',
};
```

If a value can be instantiated more than once over the lifetime of the program (e.g. a local variable declared within a function, or a static field on a class nested in a function) then it *must* use `lowerCamelCase`.

```
  private static readonly MY_SPECIAL_NUMBER = 5;

  bar() {
    return 2 * Foo.MY_SPECIAL_NUMBER;
  }
}
```

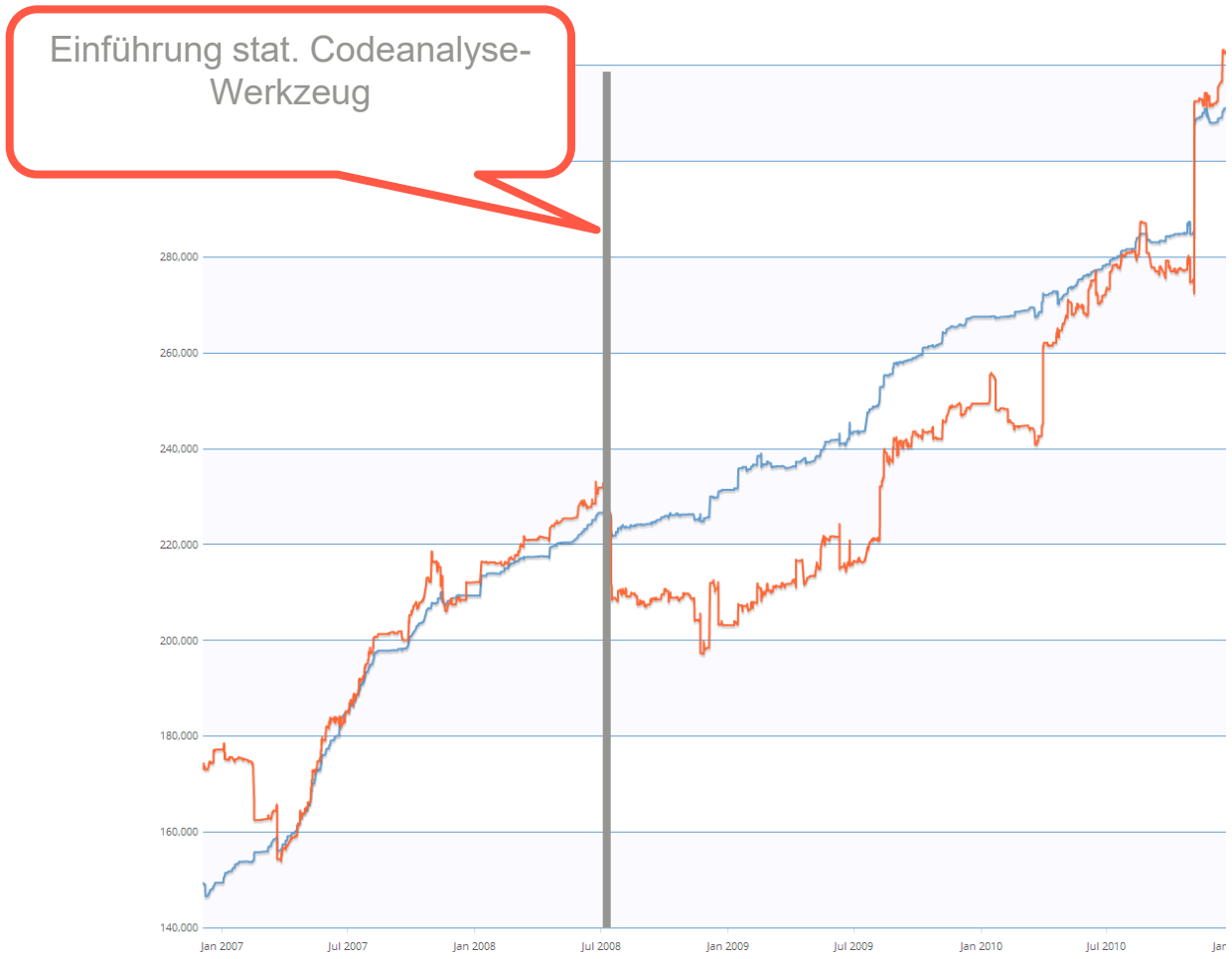
Global: Only symbols declared on the module level, static fields of module level classes, and values of module level enums, *may* use `CONST_CASE`. If a value can be instantiated more than once over the lifetime of the program (e.g. a local variable declared within a function, or a static field on a class nested in a function) then it *must* use `lowerCamelCase`.

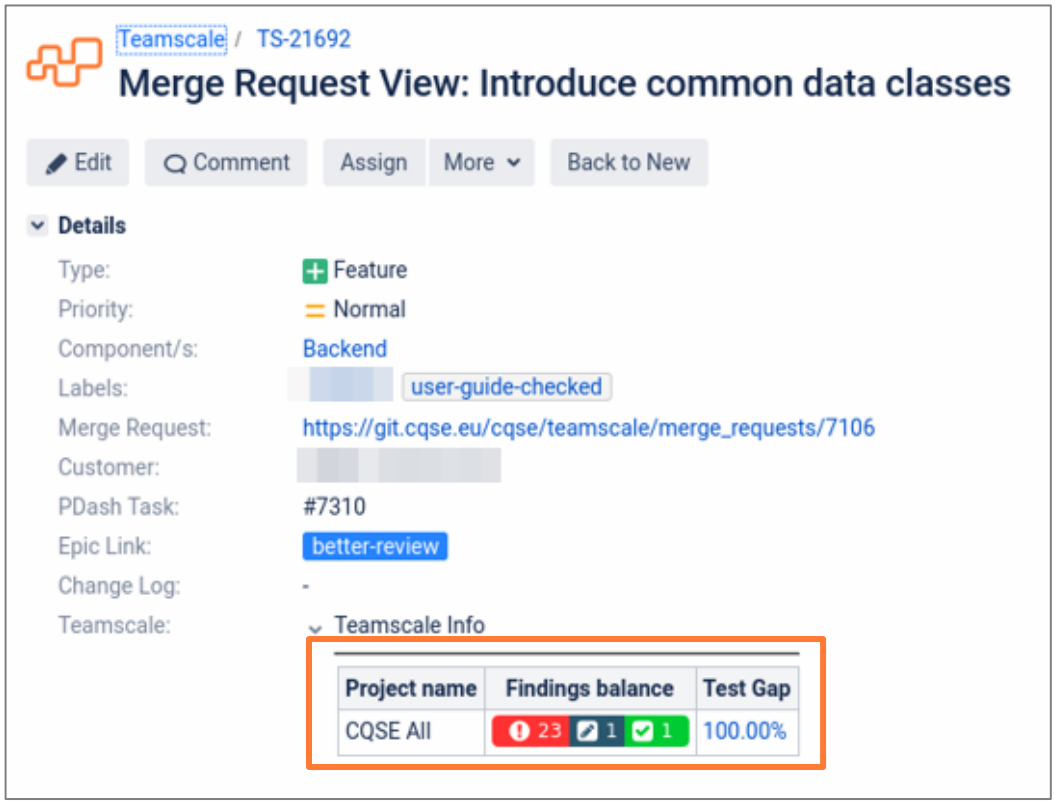
If a value is an arrow function that implements an interface, then it *may* be declared `lowerCamelCase`.

Quelle: Google, TypeScript Style Guide, Abschnitt „Constants“, CC BY 3.0 – https://google.github.io/styleguide/tsguide.html

Teamscale by CQSE parcIT

Keine langfristige Besserung



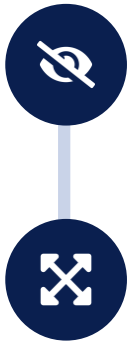


Wie es bei uns dazu kam



Lange Jahre SonarQube Community Edition. Findings da, aber gut ignorierbar.

Wie es bei uns dazu kam



Lange Jahre SonarQube Community Edition. Findings da, aber gut ignorierbar.

Das Monorepo wächst 13 Millionen Zeilen Code. Die Community Edition reicht nicht mehr.

Teamscale by CQSE parcIT

Wie es bei uns dazu kam

Lange Jahre SonarQube Community Edition. Findings da, aber gut ignorierbar.

Das Monorepo wächst 13 Millionen Zeilen Code. Die Community Edition reicht nicht mehr.

Migration Umstieg auf Teamscale. Und plötzlich sind die Findings präsent.

Wie es bei uns dazu kam

Lange Jahre SonarQube Community Edition. Findings da, aber gut ignorierbar.

Das Monorepo wächst 13 Millionen Zeilen Code. Die Community Edition reicht nicht mehr.

Migration Umstieg auf Teamscale. Und plötzlich sind die Findings präsent.

Die Frage kommt auf Wie sind diese Regeln entstanden und ergeben sie überhaupt Sinn?

Wie es bei uns dazu kam

Lange Jahre SonarQube Community Edition. Findings da, aber gut ignorierbar.

Das Monorepo wächst 13 Millionen Zeilen Code. Die Community Edition reicht nicht mehr.

Migration Umstieg auf Teamscale. Und plötzlich sind die Findings präsent.

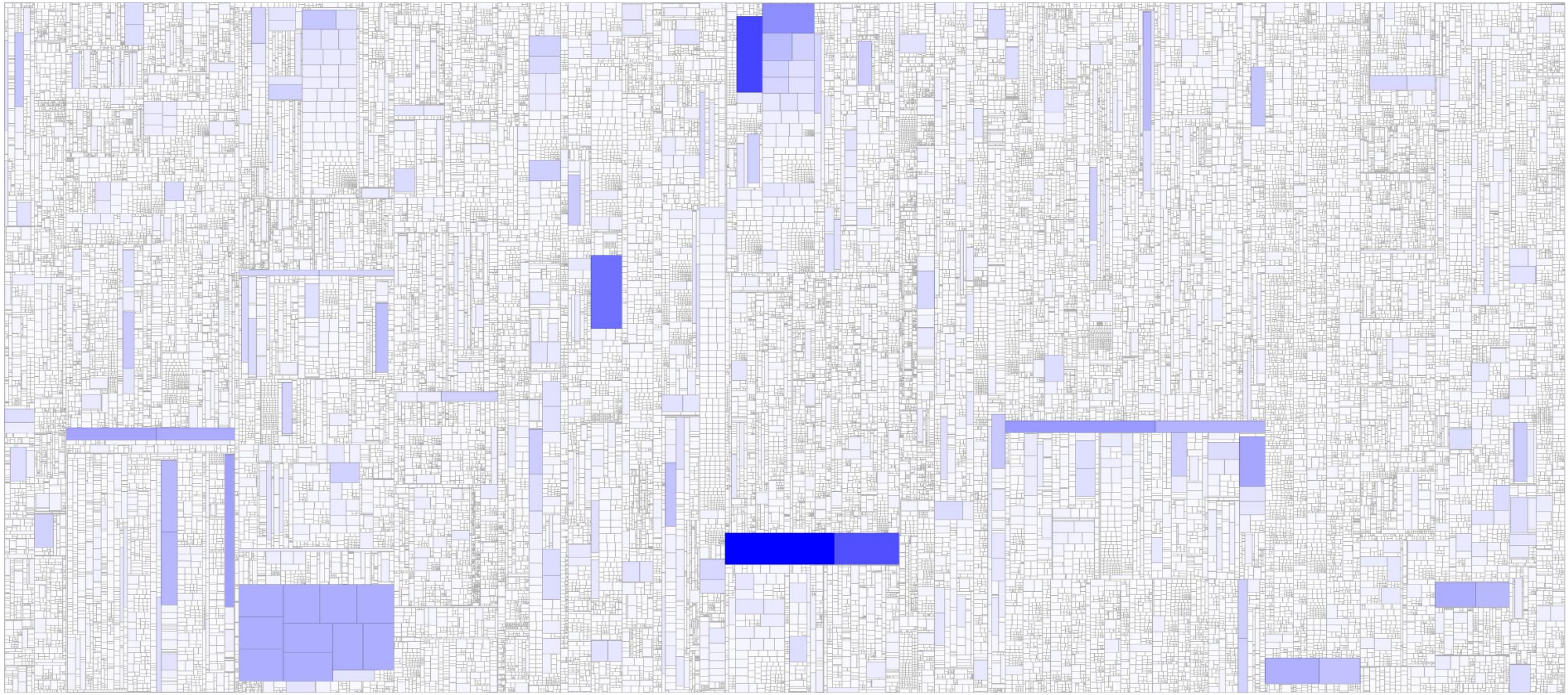
Die Frage kommt auf Wie sind diese Regeln entstanden und ergeben sie überhaupt Sinn?

Die ehrliche Antwort: Wenige Personen haben nach bestem Wissen entschieden

Das ist unser Monorepo

Jedes Rechteck eine Datei

13 Mio. Zeilen



Keine Regeldiktatur durch die QS, sondern lieber: Lasst die Teams selbst entscheiden.

„Dann schalten die doch alles ab, was weh tut.“



Warum nicht jedes Team für sich ein Profil hat

Grund 1: Wir werden behandelt wie eine Bank

Was gefordert ist

- Ein nachvollziehbares Verfahren
- Dokumentierte Entscheidungen
- Nachweisbare Zuständigkeit

„… Durchführung angemessener Tests, wie etwa […] Quellcodeprüfungen soweit durchführbar …“

Art. 25 Abs. 1 Verordnung (EU) 2022/2554 (DORA - Digital Operational Resilience Act)

Was NICHT gefordert ist

- Regel X
- Regel Y
- Irgendeine konkrete regel

Die Aufsicht will wissen, wie wir **bewusst Entscheidungen treffen**, nicht was **wir entscheiden**

Warum nicht jedes Team für sich ein Profil hat

Grund 1: Wir werden behandelt wie eine Bank

Was gefordert ist

- Ein nachvollziehbares Verfahren
- Dokumentierte Entscheidungen
- Nachweisbare Zuständigkeit

„… Durchführung angemessener Tests, wie etwa […] Quellcodeprüfungen soweit durchführbar …“

Art. 25 Abs. 1 Verordnung (EU) 2022/2554 (DORA - Digital Operational Resilience Act)

Was NICHT gefordert ist

- Regel X
- Regel Y
- Irgendeine konkrete regel

Die Aufsicht will wissen, wie wir **bewusst Entscheidungen treffen**, nicht was **wir entscheiden**

Erklärt der Aufsicht mal, warum in Team A andere Qualitätsstandards gelten als in Team B

Warum nicht jedes Team für sich ein Profil hat

Grund 2: Wir sind aus dem genossenschaftlichen Verbund entstanden



Was einer alleine nicht schafft,

das schaffen viele

Teamscale by CQSE parcIT

Warum nicht jedes Team für sich ein Profil hat

Grund 2: Wir sind aus dem genossenschaftlichen Verbund entstanden



Was einer alleine nicht schafft,
das schaffen viele



Die manchmal unbequeme Seite

Viele Gremien. Viele Meinungen.
Alles dauert länger, als man gerne hätte

Teamscale by CQSE parcIT

Warum nicht jedes Team für sich ein Profil hat

Grund 2: Wir sind aus dem genossenschaftlichen Verbund entstanden



Was einer alleine nicht schafft,
das schaffen viele



Die manchmal unbequeme Seite

Viele Gremien. Viele Meinungen.
Alles dauert länger, als man gerne hätte



Die andere Seite

Aber am Ende stehen die Institute dahinter.
Nicht nur formal

Jetzt habe ich das böse Wort gesagt: Prozess…

Warum nicht jedes Team für sich ein Profil hat

Woher der Prozess wirklich kommt

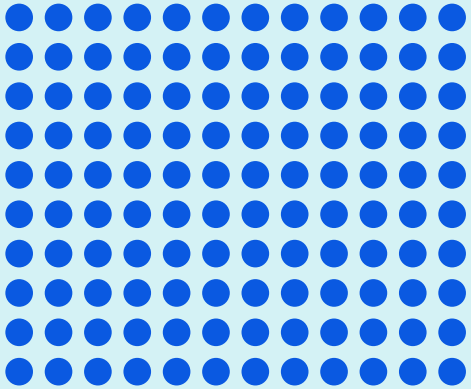
Fearless Change

Autor:innen: Manns & Rising

Meine Erkenntnis: Veränderung gelingt nicht, wenn sie verkündet wird, sondern wenn möglichst viele mitgenommen werden.

120

Entwickler:innen, die jeden Tag mit diesen Regeln konfrontiert werden, ob sie wollen oder nicht.



Warum nicht jedes Team für sich ein Profil hat

Woher der Prozess wirklich kommt

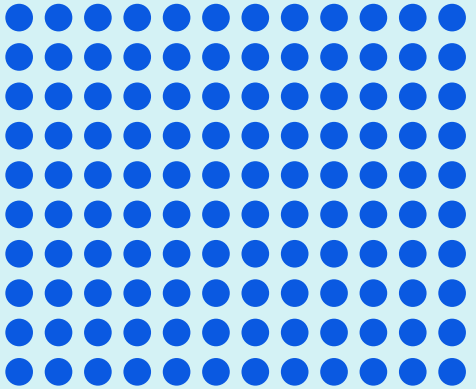
Fearless Change

Autor:innen: Manns & Rising

Meine Erkenntnis: Veränderung gelingt nicht, wenn sie verkündet wird, sondern wenn möglichst viele mitgenommen werden.

120

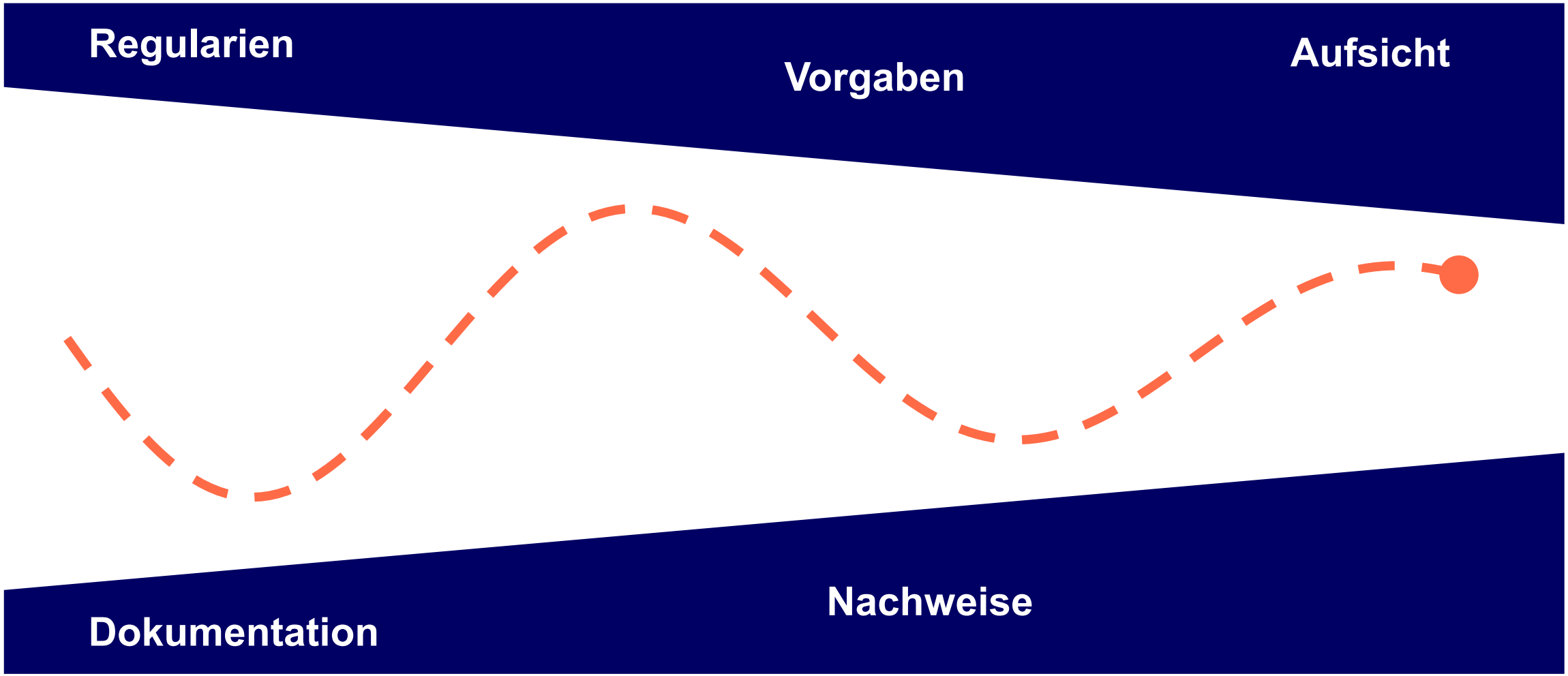
Entwickler:innen, die jeden Tag mit diesen Regeln konfrontiert werden, ob sie wollen oder nicht.



„Ich werde gehört“ ist kein Wohlfühlfaktor.
Es ist der Grund, warum die Regeln nachher nicht ignoriert werden.

Warum mir das wichtig war

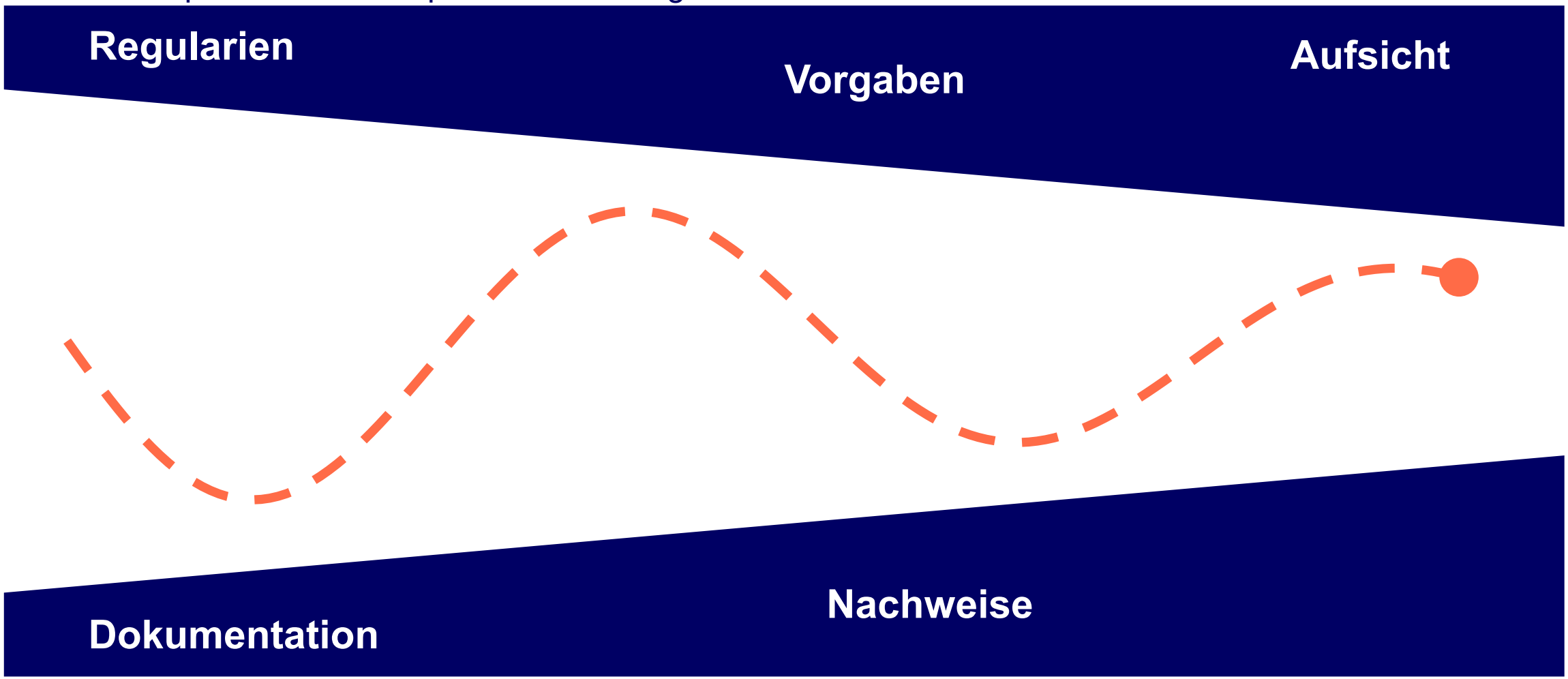
Sechs Jahre parcIT und der Spielraum wird enger



Teamscale by CQSE parcIT

Warum mir das wichtig war

Sechs Jahre parcIT und der Spielraum wird enger



Die Mauern wählen wir nicht. Den Weg dazwischen schon

Wie ein Antrag beginnt

Was es als Erstes braucht

Ein Ticket 

1. **Um welche Regel geht es?** Inklusive der Original-Description

Teamscale by CQSE parcIT

Wie ein Antrag beginnt

Was es als Erstes braucht

Ein Ticket 

1. **Um welche Regel geht es?** Inklusive der Original-Description
2. **Was soll sich ändern?** Aktivieren, abschalten, anpassen, Code-Scope: Test- oder Prod-Code

Teamscale by CQSE parcIT

Wie ein Antrag beginnt

Was es als Erstes braucht

Ein Ticket 

1. **Um welche Regel geht es?** Inklusive der Original-Description
2. **Was soll sich ändern?** Aktivieren, abschalten, anpassen, Code-Scope: Test- oder Prod-Code
3. **Und warum?** Der einzige Teil, der wirklich Arbeit macht

„Weil ich das schon immer so gemacht habe“, lasse ich nicht gelten

Teamscale by CQSE parcIT

Wie ein Antrag beginnt

Der Antrag

Teamscale Regel

Comprehensibility › Bad Practice › Usage of the var keyword

„The usage of the 'var' keyword is discouraged as it makes the code harder to read, because the result type is not always immediately apparent from the right-hand side, e.g. result method invocations. It can lead to error-prone code throughout the following usage of the variable.“

Der Antrag lautet: **Regel AKTIVIEREN, daher var faktisch verbieten**

Ein Team will eine Regel hinzunehmen und nicht abschalten

QS-2072 - Comprehensibility > Bad Practice > Usage of the var keyword

Findings Filter auf Staging

Avoid usage of the var keyword.

Why is This a Problem?

The usage of the 'var' keyword is discouraged as it makes the code harder to read, because the result type is not always immediately apparent from the right-hand side, e.g. result method invocations. It can lead to error-prone code throughout the following usage of the variable.

Exceptions

It can be justifiable for statements where the right hand side is an object allocation.

```
var list = new ArrayList<String>();
```

How Can I Resolve This?

Explicitly state the type of the variable.

Example (Before)

```
var x = "foo";
```

Example (After)

```
String x = "foo";
var x = new String("foo");
```

Begründung

GBS:

> Siehe Beschreibung:

> The usage of the 'var' keyword is discouraged as it makes the code harder to read, because the result type is not always immediately apparent from the right-hand side, e.g. result method invocations. It can lead to error-prone code throughout the following usage of the variable.

[redacted] excel.FilterTableExcelExport.java

The var keyword should not be used

```
295                     ++rowNumber;
296                 }
297             }
298         }
299
300         private void markCellsWithDisabledCriterion(
301             Map<P, IntRange> rowRangeByPosition,
302             Map<C, Integer> columnByCriterion,
303             CellStyle disabledCellStyle) {
304
305             for (var rowRangeForPosition : rowRangeByPosition.entrySet()) {
306                 for (var columnRangeForCriterion : columnByCriterion.entrySet()) {
307                     if (isCriterionEnabled(rowRangeForPosition.getKey(), columnRangeForCriterion.getKey())) {
308                         continue;
309                     }
310
311                     for (int rowNumber : rowRangeForPosition.getValue()) {
312                         XSSFRow row = getOrCreateRow(getTable(), rowNumber);
313                         int columnNumber = unbox(columnRangeForCriterion.getValue());
314
315                         getOrCreateCell(row, columnNumber).setCellStyle(disabledCellStyle);
```

Map.Entry<P, IntRange> rowRAngeForPosition : rowRangeByPosition.entrySet()

[redacted] Adapter.java

The var keyword should not be used

```
17
18          [redacted]
19
20              [redacted]
21              [redacted]
22          }
23
24          private static final Supplier<[redacted]> map[redacted]( ResolverAPI resolver) {
25
26              return () -> {
27                  final var resolve = resolver.resolve();
28                  return switch (resolve.get()) {
29                      case [redacted]
30                      case [redacted]
31                      case [redacted]
32                      case [redacted]
33                      case [redacted]
34                      case [redacted]
35                      case [redacted]
36                      case [redacted]
37                  };
```

Jetzt seid ihr die Expertengruppe

Runde 1: Wie würdet ihr entscheiden

var hilft

```
var url = new URL("https://…");
var conn = url.openConnection();
var reader = new BufferedReader(
    new InputStreamReader(
        conn.getInputStream()));
```

var verschluckt informationen

```
var result = service.calculate(id);

// Welchen Typ hat result?
// Und was passiert 40 Zeilen weiter
unten damit?
```

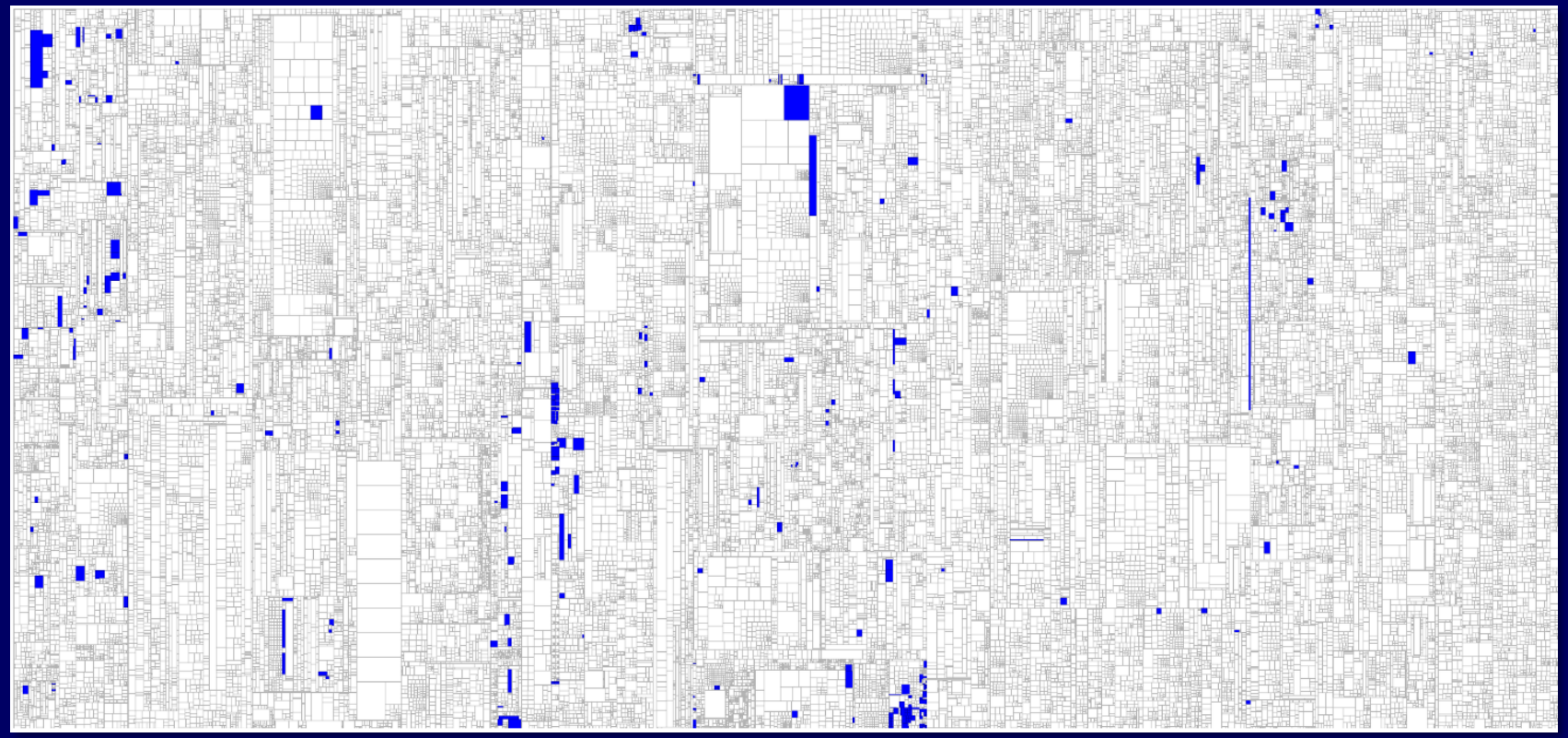
Stimmt jetzt ab: Soll die Regel aktiviert werden?

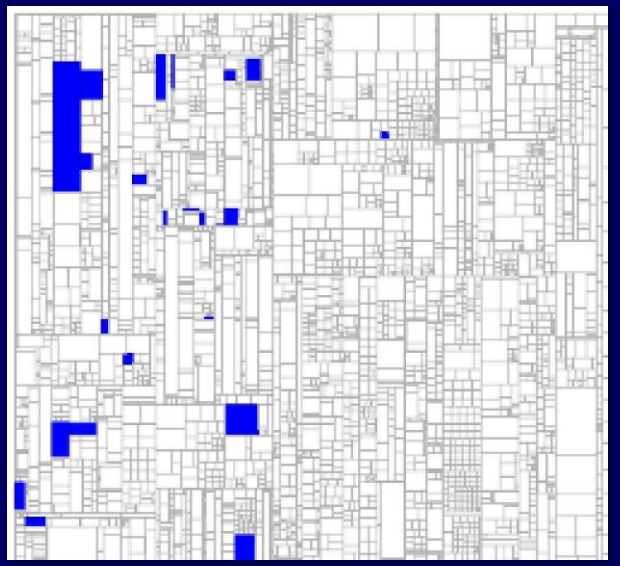


Teamscale by CQSE parcIT

Jetzt seid ihr die Expertengruppe

Findings-Verteilung: var-Usage





Die Regel hätte fast nichts gekostet.

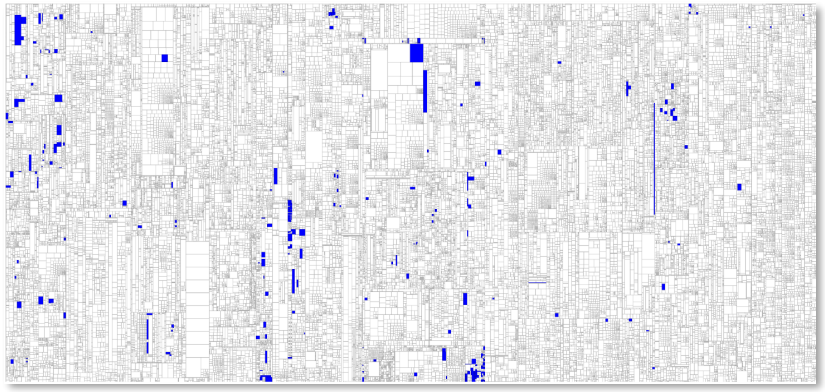
Die Aktivierung hätte kaum neue Findings erzeugt

Teamscale by CQSE parcIT

menti.com: **4330 6068**

Jetzt seid ihr die Expertengruppe

Runde 2: Jetzt mit Daten



Stimmt jetzt ab: Gleiche Frage. Jetzt mit Daten

Wie entschieden wird

Die Expertengruppe



11 Teams

Jedes schickt einen Vertreter

Einmal im Monat

2 min

Um welche Regel geht es?

Die Vertreter:innen des antragstellenden Teams begründet die Anpassung.

Wie entschieden wird

Die Expertengruppe



11 Teams

Jedes schickt einen Vertreter

Einmal im Monat

2 min

Um welche Regel geht es?

Die Vertreter:innen des antragstellenden Teams begründet die Anpassung.

15 min

Diskussion

Handzeichen der Reihe nach. Moderierte Diskussion. Einmalig um 5 Minuten verlängerbar

Wie entschieden wird

Die Expertengruppe



11 Teams

Jedes schickt einen Vertreter

Einmal im Monat

2 min

Um welche Regel geht es?

Die Vertreter:innen des antragstellenden Teams begründet die Anpassung.

15 min

Diskussion

Handzeichen der Reihe nach. Moderierte Diskussion. Einmalig um 5 Minuten verlängerbar

1 min

Abstimmung · Antrag Angenommen ab 60% Zustimmung

Namentlich. Ergebnis wird dokumentiert, inklusive wer wir für welches Team gestimmt hat

Feste Agenda. Es kommt immer zu einer Entscheidung

Wie entschieden wird

Die unbequemste Regel im Prozess

Ist kein Vertreter des antragstellenden Teams anwesend, gilt die Regelanpassung als abgelehnt.



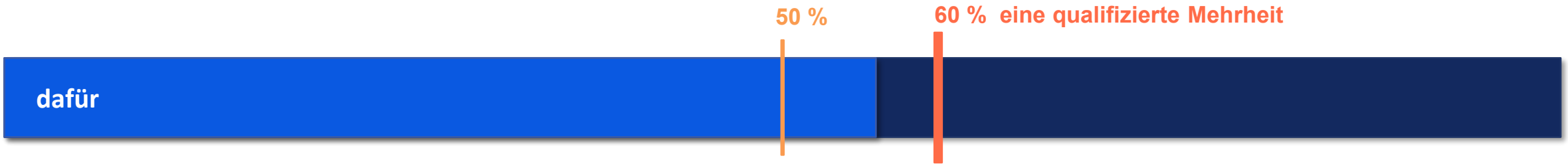
Wer den Antrag stellt, sitzt auch im Raum. Wer die Diskussion anzettelt und dann nicht auftaucht, lässt die anderen mit seinem Problem allein

Keine Schikane, sondern Eintrittskarte in die Mitbestimmung

Teamscale by CQSE parcIT

Wie der Fall ausging

Unser Ergebnis in der parcIT



Die Mehrheit

war dafür

Aber nicht 60 %

Abgelehnt

**„Die Frage war am Ende nicht, was die Regel kostet.
Sondern ob man sich ein Sprachmittel dauerhaft wegnehmen lässt.“**

Wie der Fall ausging

Nicht mal die Erfinder sind sich einig



„There is a certain amount of controversy over this feature. Some welcome the concision it enables; others fear that it deprives readers of important type information, impairing readability.

And both groups are right.”

Stuart W. Marks · Local Variable Type Inference: Style Guidelines · OpenJDK, Project Amber

„Wenn nicht mal die Leute, die das Feature gebaut haben, eine eindeutige Antwort geben, warum sollte sie dann in einem Default-Regelset stehen?“

openjdk.org/projects/amber/guides/lvti-style-guide

Teamscale by CQSE parcIT

Zwei Jahre später

Und diese in Zahlen

45

Regelanpassungs-Anträge insgesamt

5

davon abgelehnt

1

Monat maximal vom Ticket zur Entscheidung

11

Teams mit je einem Vertreter im Gremium

Zwei Jahre später

Und diese in Zahlen

45

Regelanpassungs-Anträge insgesamt

5

davon abgelehnt

1

Monat maximal vom Ticket zur Entscheidung

11

Teams mit je einem Vertreter im Gremium



Was mich überrascht hat: die Einigkeit

Ich hatte mich auf Grabenkämpfe und Eskalationen eingestellt. Eskalationsverfahren habe ich nie gebraucht. Die Kolleg:innen waren sich meistens einig.

Zwei Jahre später

Die ehrliche Folie

13 Millionen Zeilen

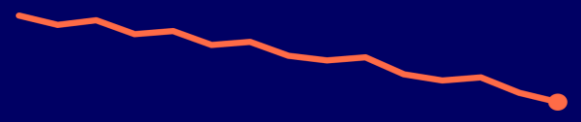
Altlasten aus Jahrzehnten.
Daran ändert kein Prozess der Welt etwas über Nacht

… nicht perfekt

Viele offene Findings die es noch zu beheben gilt

Jedoch, die Kurve stimmt

Viele offene Findings die es noch zu beheben gilt



Findings-Tendenz, schematisch

Zwei Jahre später

Die ehrliche Folie

13 Millionen Zeilen

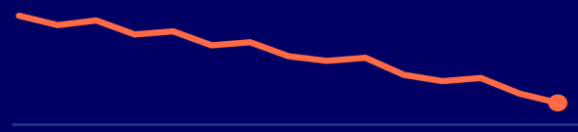
Altlasten aus Jahrzehnten.
Daran ändert kein Prozess der Welt etwas über Nacht

… nicht perfekt

Viele offene Findings die es noch zu beheben gilt

Jedoch, die Kurve stimmt

Viele offene Findings die es noch zu beheben gilt



Findings-Tendenz, schematisch

„Man wird nicht von heute auf morgen gut
Solange die Tendenz stimmt, wird es jeden Tag ein bisschen besser

Zwei Jahre später

Erinnert ihr euch an die Befürchtung von vorhin?

„Dann schalten die doch alles ab, was weh tut.“



Heute kommen mehr Anträge, Regeln hinzuzunehmen, als welche abzuschalten.

Die Grenzen

Wo unsere Demokratie aufhört

Die nicht-verhandelbare Zone

Programmier- und Security-Richtlinien, bei denen die Runde keinen Einfluss hat.

Auch dort sitzen Fachleute. Sie müssen nur die unangenehmeren Entscheidungen treffen.

Unsere Leitlinie dabei

So viel wie nötig, so wenig wie möglich.

Jede Regel, die dort landet, fehlt der Gruppe, daher landen dort nur die, die wirklich müssen.

Was bleibt

Am Anfang stand eine Frage

Wer hat diese Regel eigentlich aktiviert



Was bleibt

Am Anfang stand eine Frage

Wer hat diese Regel eigentlich aktiviert

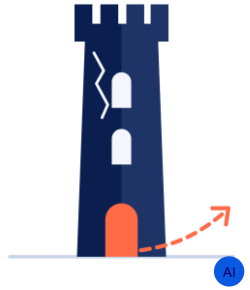


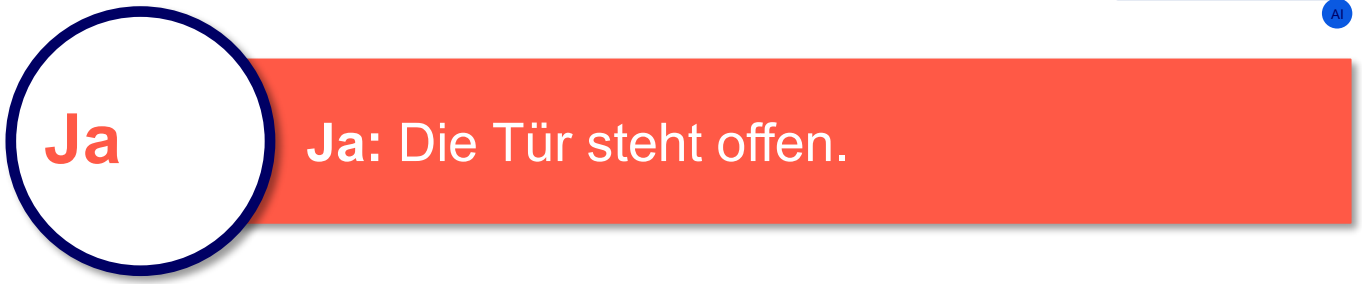
Bei uns kann die heute jeder beantworten.
Es steht im Protokoll: mit Namen, Entscheidungen und Aktivierungszeitpunkt

Teamscale by CQSE parcIT

Was bleibt

Haben wir den Elfenbeinturm abgerissen?





Was bleibt

Haben wir den Elfenbeinturm abgerissen?



Nein: Der Turm steht noch.

Jein

Ja: Die Tür steht offen.

Was geblieben ist

Es entscheidet immer noch ein Gremium für alle. Das verlangt unser Kontext, und daran ändern wir nichts.

Was sich geändert hat

Vorher entschied eine isolierte Gruppe. Wer Einfluss wollte, musste lobbyieren gehen.

Wir haben das Lobbyieren durch ein Wahlrecht ersetzt.

Was bleibt

Ach! Bevor ich es vergesse:



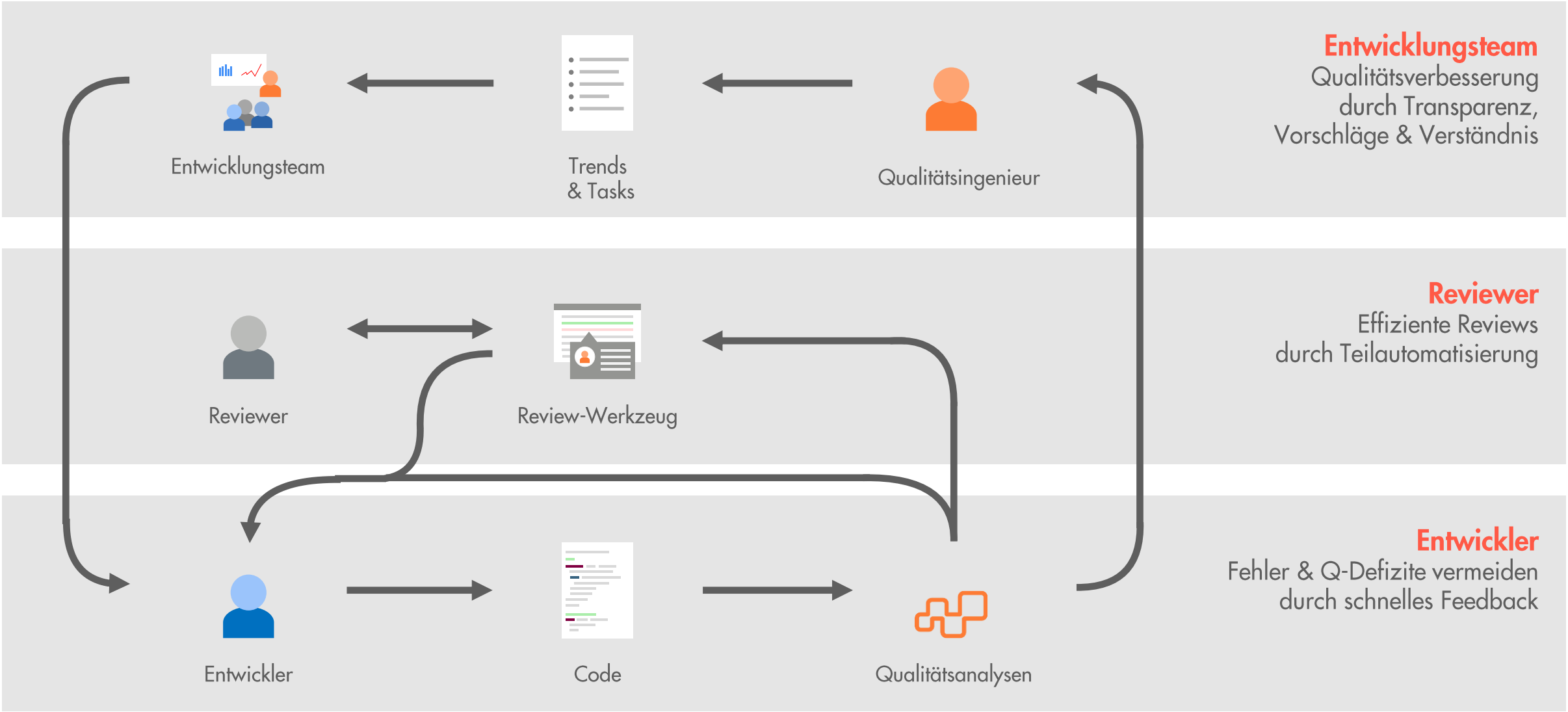
Geprüft. Keine Mängel.

Interne Revision und externe Wirtschaftsprüfer haben sich den Prozess angesehen.

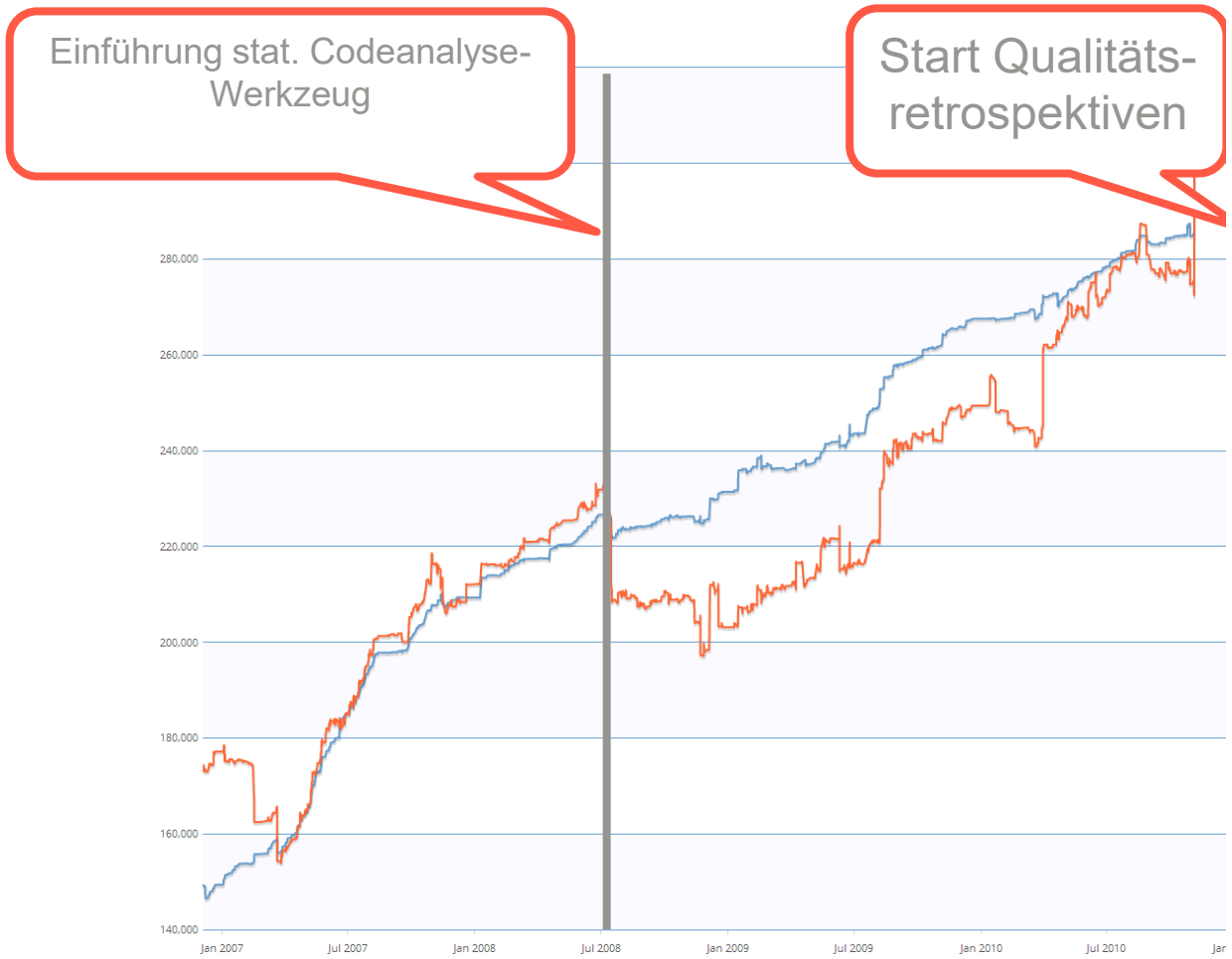
Ein Lob gibt es dafür übrigens nie…

Aber darum ging es auch nie.
Es ging um einen Prozess, der versucht, 120 Entwickler:innen
und ihre Bedürfnisse zu sehen.

Langfristig Qualität sichern



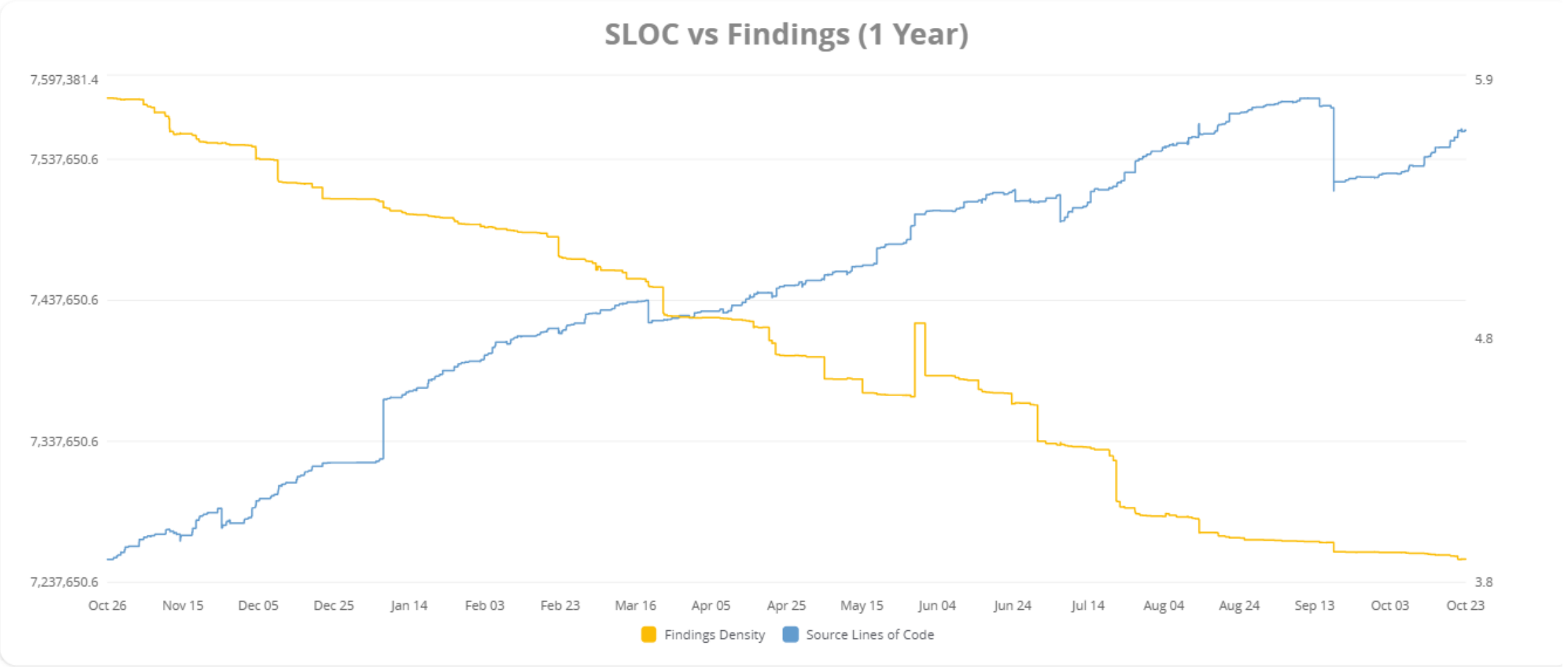
Langfristig Qualität sichern



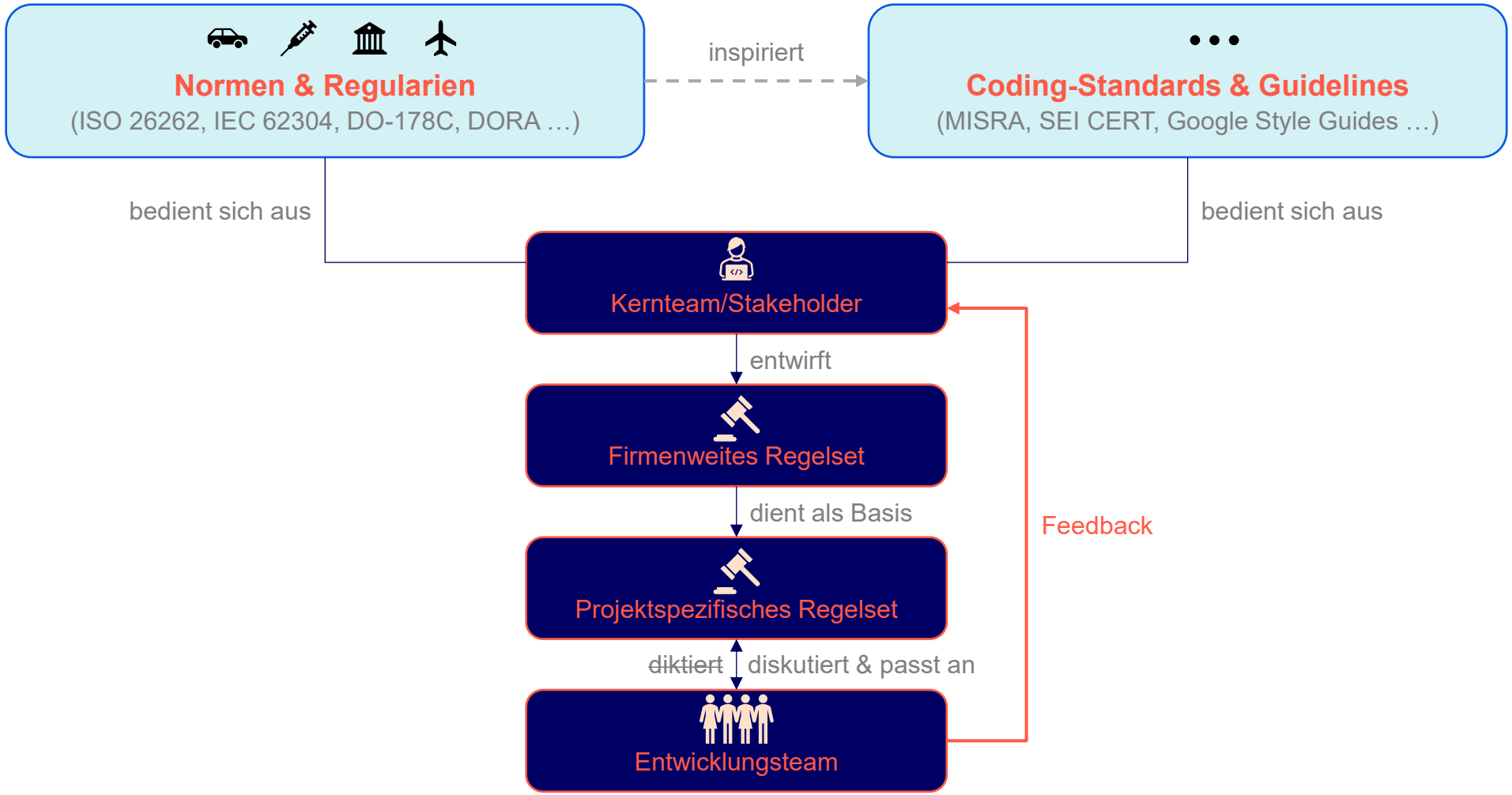
→ Prozesse und Retrospektiven führen zu langfristigen Qualitätsverbesserungen

Teamscale by CQSE parcIT

Impression ParcIT



Wie entstehen gute Regelsets in Unternehmen?



Was wir euch mitgeben wollen

KI braucht Leitplanken

1

Statische Codeanalyse ist in Zeiten von KI so wichtig wie noch nie

Codequalität als Teamsport

2

Ein Tool alleine ist meistens noch nicht der ganze Weg zu nachhaltiger Codequalität

Seid offen für Diskussion

3

Überlegt euch einen Prozess, wie man gemeinsam über Programmierrichtlinien diskutieren kann

Teamscale by CQSE parcIT

Wenn ihr noch tiefer einsteigen wollt

15:15 - 16:15 Session DE

Qualität trotz kurzer Releasezyklen und KI-Einsatz – Mit Continuous Quality Control

Softwareentwicklung im Enterprise-Kontext muss branchenübergreifend immer kürzeren Releasezyklen gerecht werden und die Qualität beim Einsatz von KI sicherstellen. Wir können deshalb Releases nicht länger in dedizierten Testphasen absichern, sondern **Mehr lesen...**



Tobias Röhm, *Teamscale by CQSE*