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

W-JAX 2026

Teamscale by CQSE

Dr. Tobias Röhm (roehm@cqse.eu)



Herausforderungen

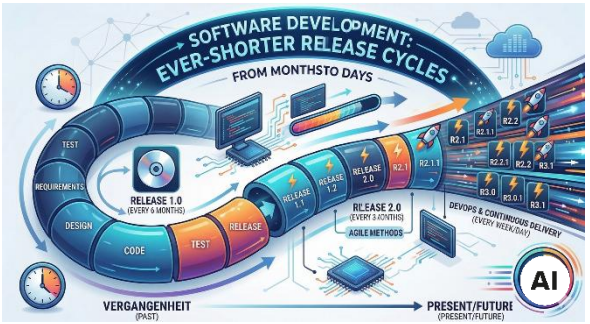
Tesla bietet Hurrikan-Opfern mehr Akku – wie geht das?



Ereignisbasierte
»Over the air«-Updates



Zeitkritische
Sicherheitspatches



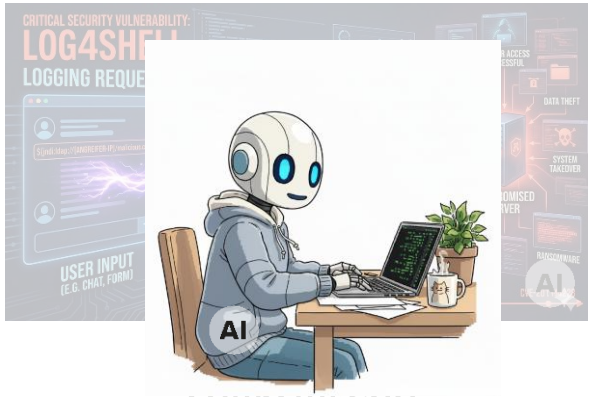
Kürzere Releasezyklen

→ Software-Releases müssen jederzeit möglich sein
→ Software muss jederzeit ein releasefähiges Qualitätsniveau aufweisen

Herausforderungen

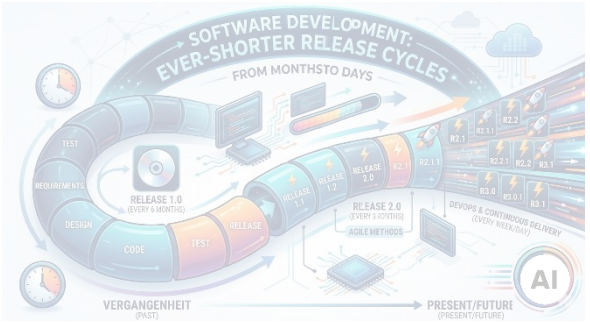


Ereignisbasierte
»Over the air«-Updates



Zeitkritische
Sicherheitspatches

Auswirkung von
Coding-Agenten?



Kürzere Releasezyklen

→ Software-Releases müssen jederzeit möglich sein
→ Software muss jederzeit ein releasefähiges Qualitätsniveau aufweisen

Herausforderungen

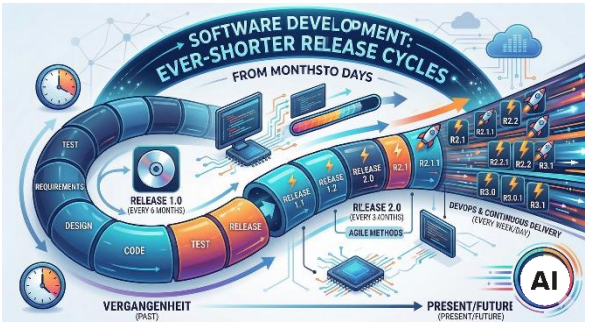
Tesla bietet Hurrikan-Opfern mehr Akku – wie geht das?



Ereignisbasierte
»Over the air«-Updates



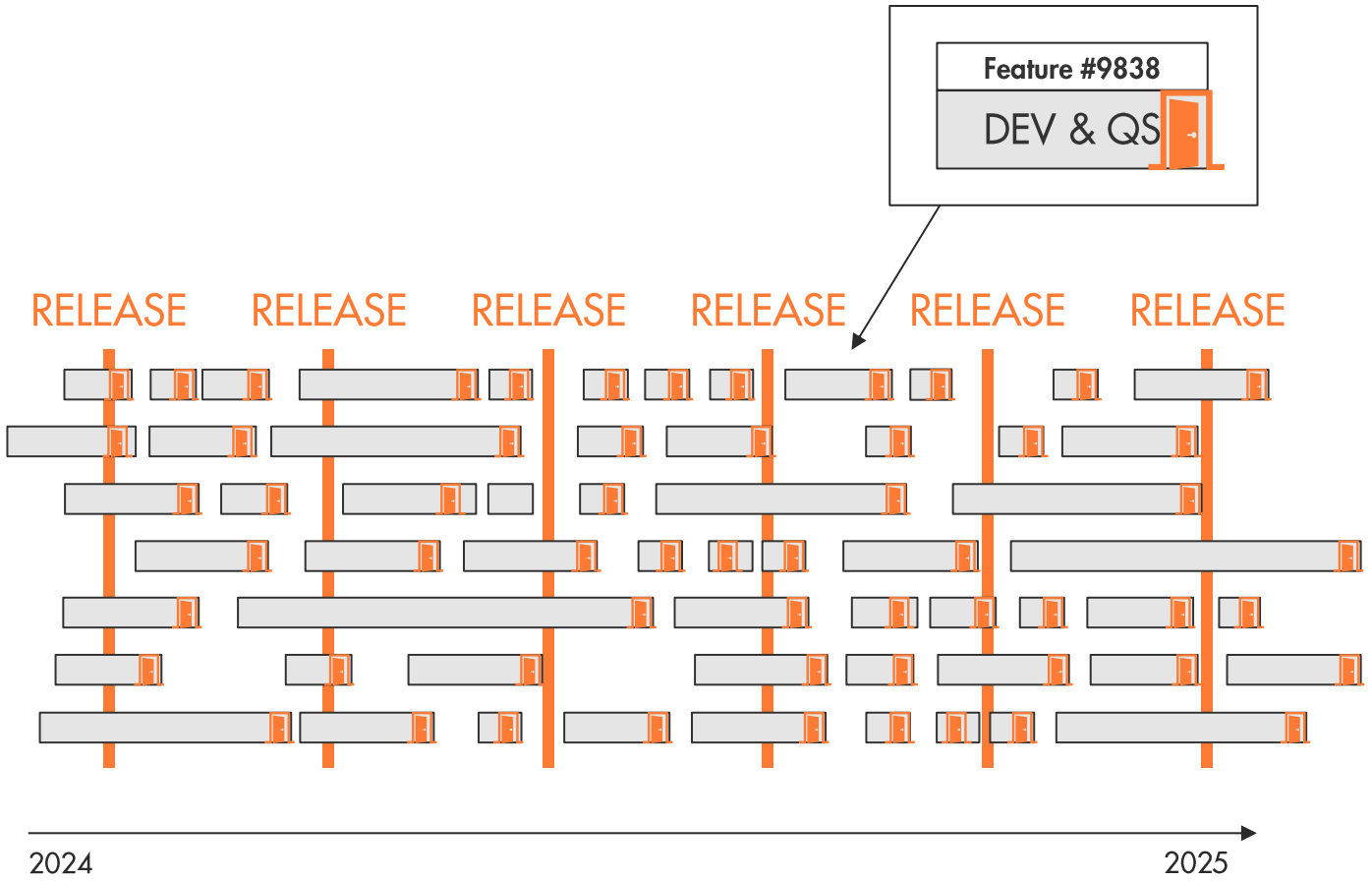
Zeitkritische
Sicherheitspatches



Kürzere Releasezyklen

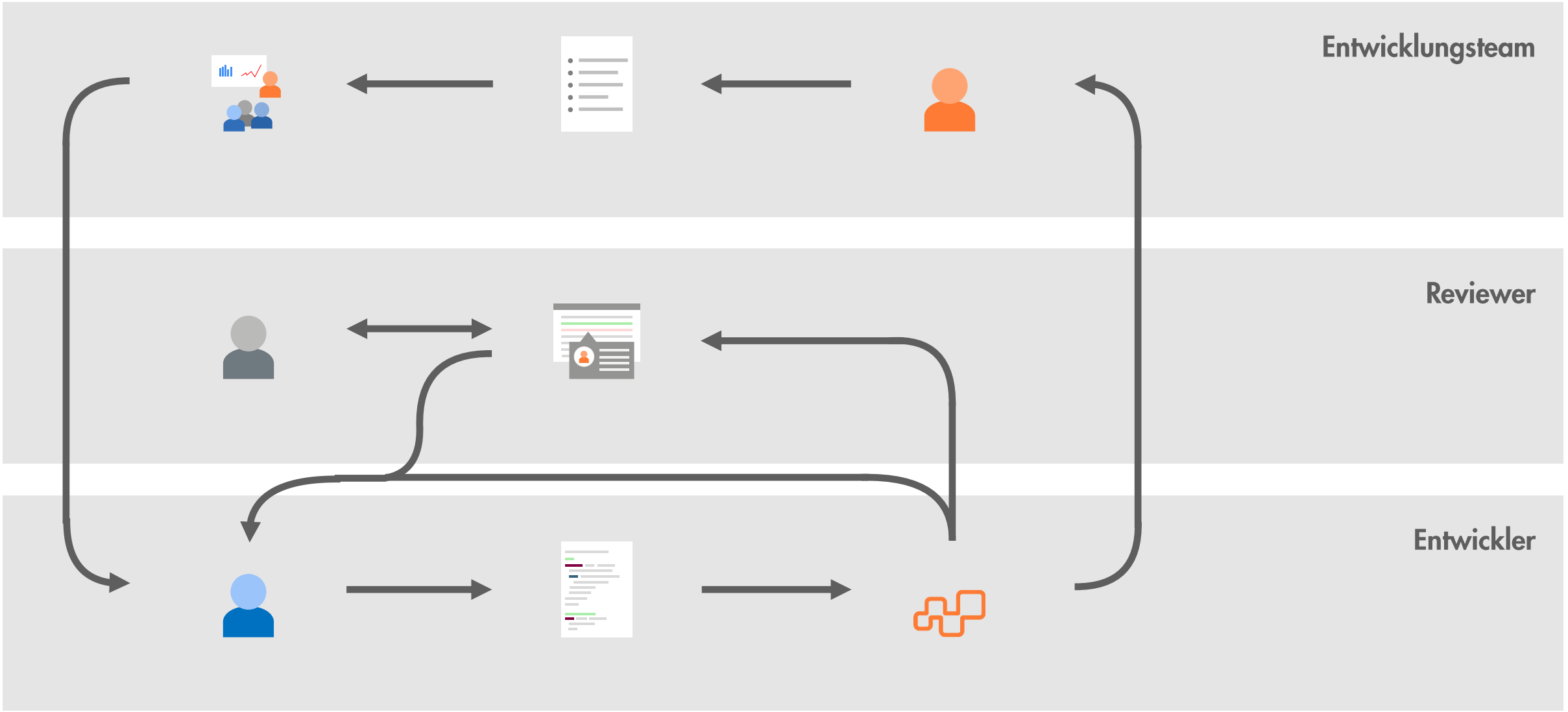
→ Software-Releases müssen jederzeit möglich sein
→ Software muss jederzeit ein releasefähiges Qualitätsniveau aufweisen

Lösung: Continuous Quality Control (CQC)

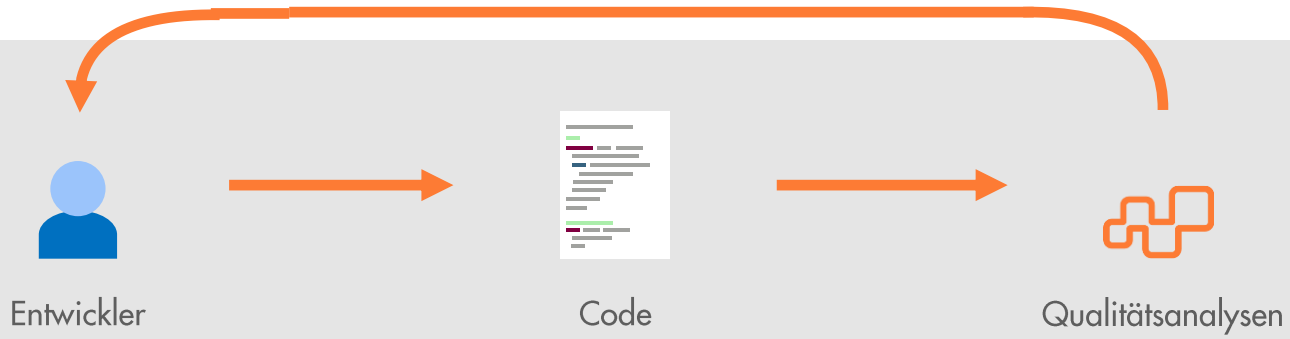


→ Pull Request als »Qualitätstürchen«
→ Qualität ist jederzeit auf releasefähigem Niveau

Lösung: Continuous Quality Control

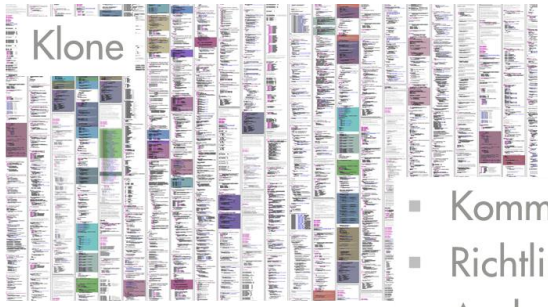


Feedbackschleife 1: Statische Codeanalyse hilft Entwicklern, Fehler & Qualitätsdefizite zu vermeiden



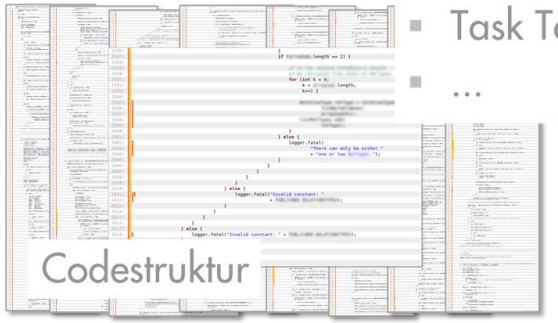
Entwickler-Feedback
Fehler & Qualitätsdefizite vermeiden
durch schnelles Feedback

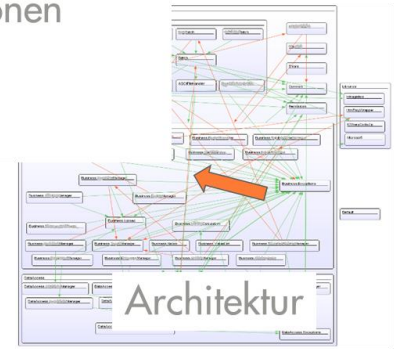
Übersicht Statische Codeanalyse

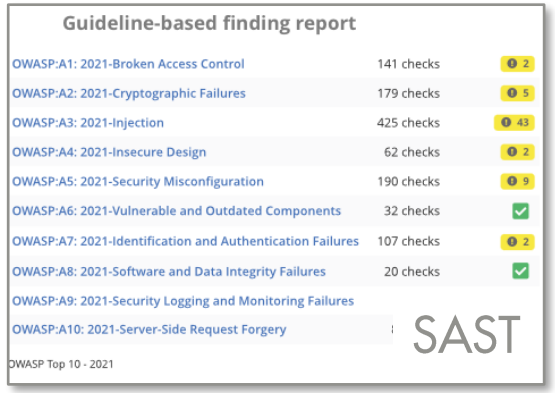




- Kommentarvollständigkeit
- Richtlinienverletzungen
- Auskommentierter Code
- Namenskonventionen
- Task Tags
- …

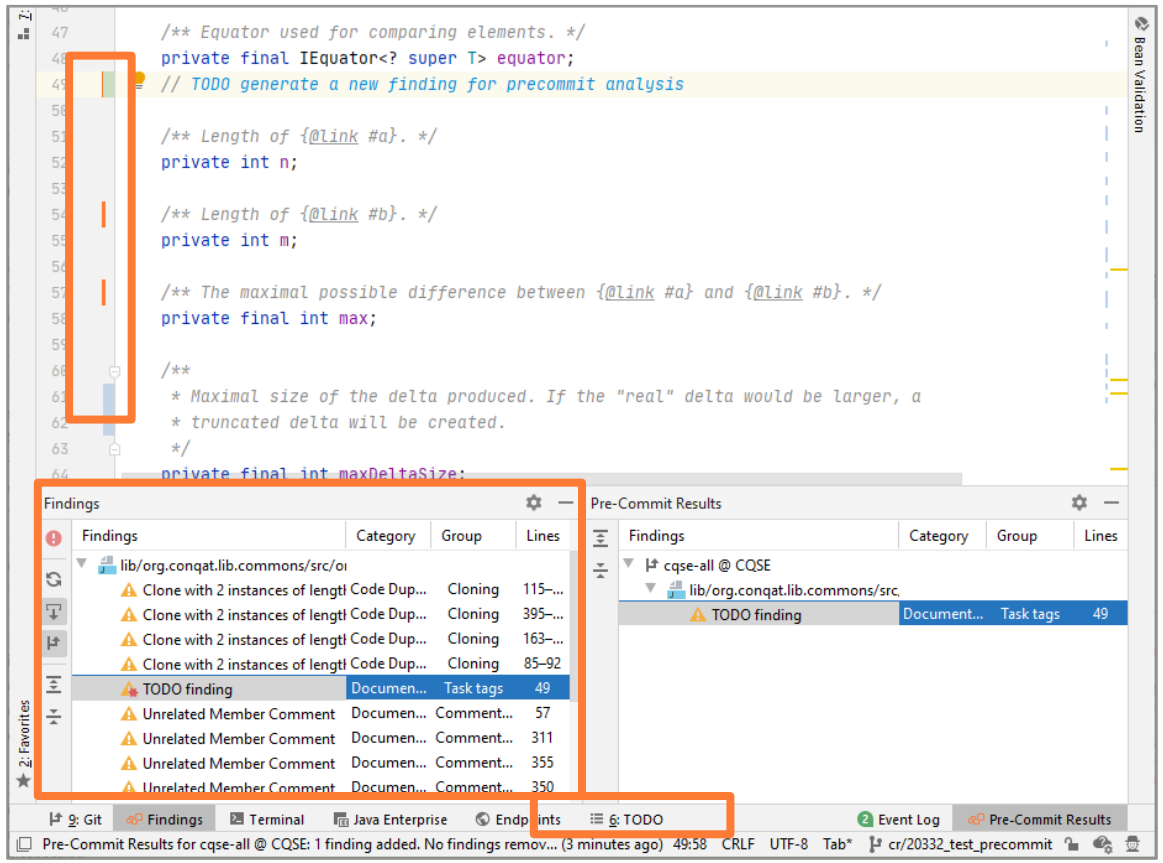






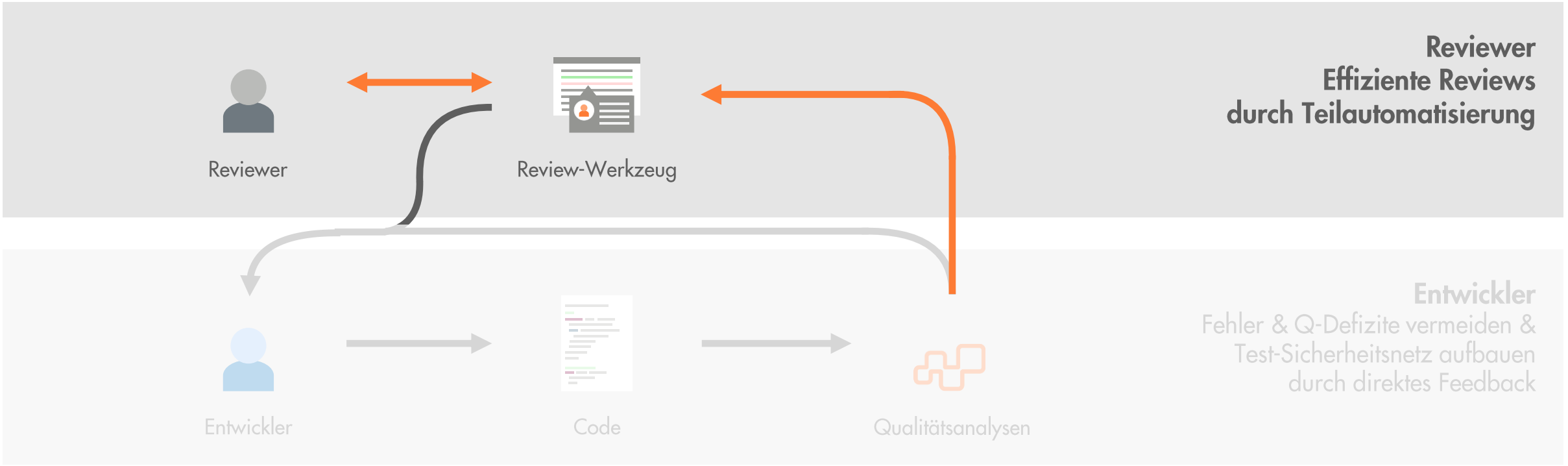
→ Statische Codeanalyse schafft Transparenz bezüglich Code- & Architekturqualität

IDE-Integration & Precommit-Analyse

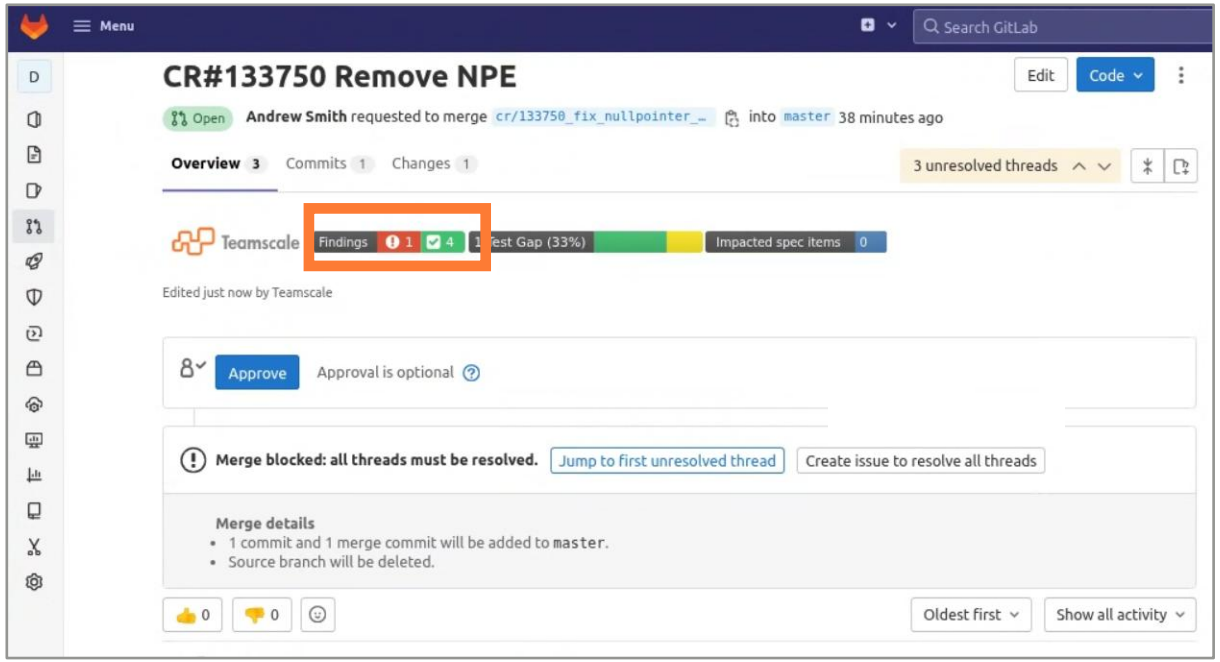


→ Feedback aus statischer Codeanalyse ist schnell, fokussiert und integriert nutzbar

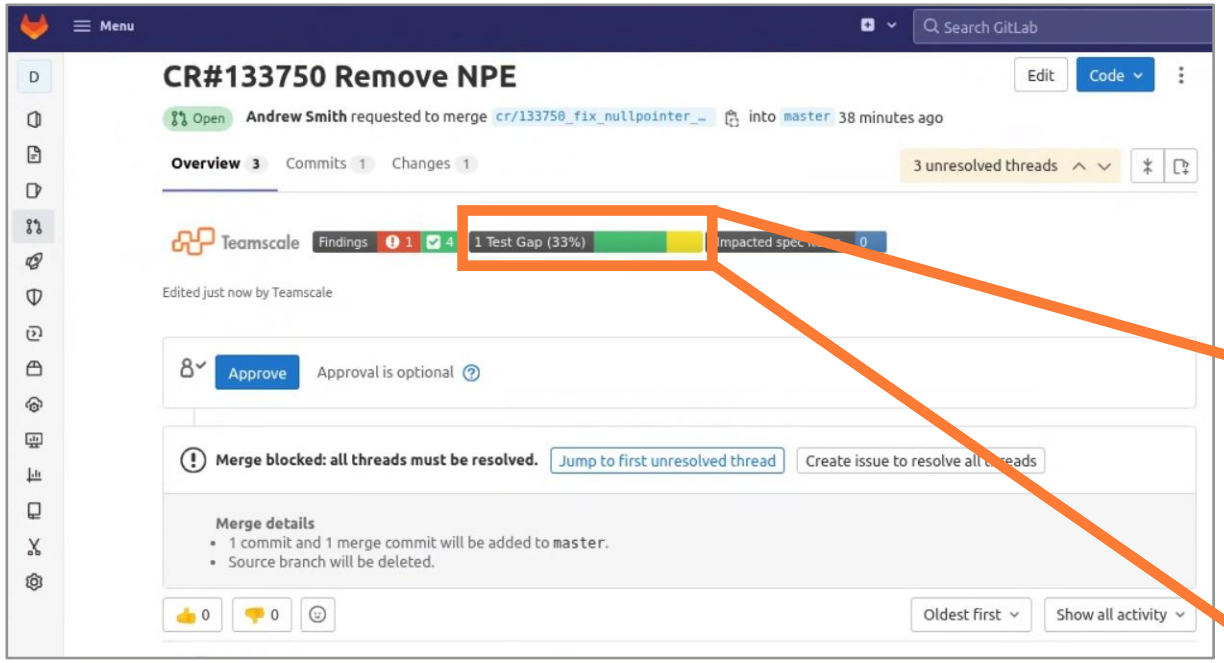
Feedbackschleife 2: Stat. Codeanalyse und Test-Gap-Analyse ermöglichen Reviewern effiziente Code Reviews

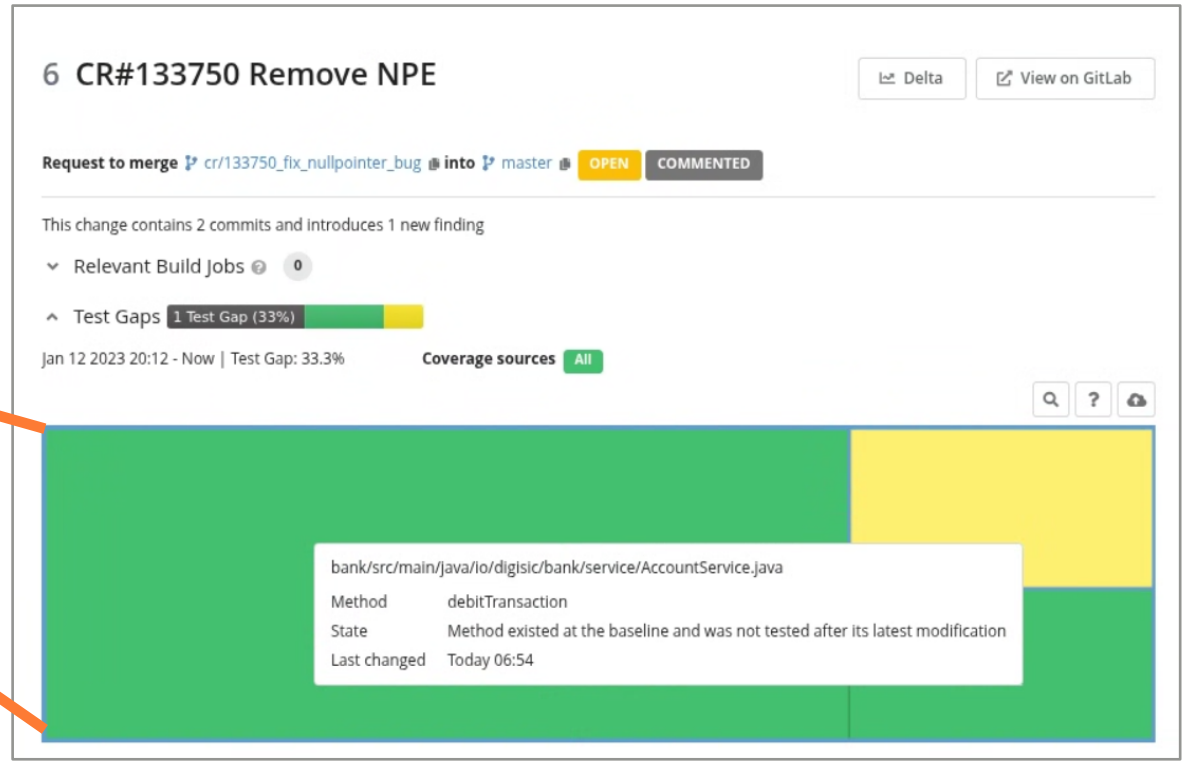


Annotation von Findings an Pull Requests

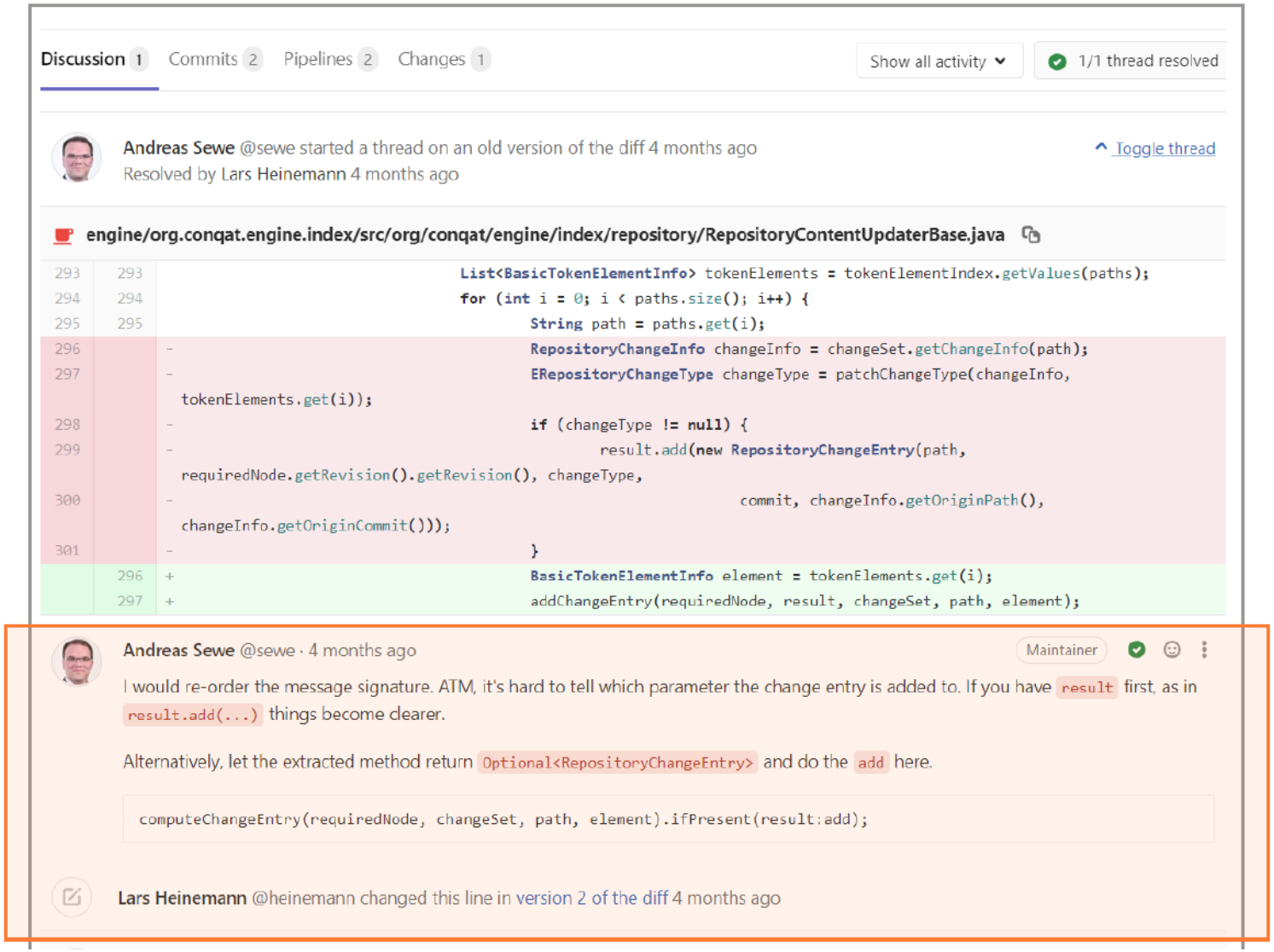


Annotation von Test Gaps an Pull Requests



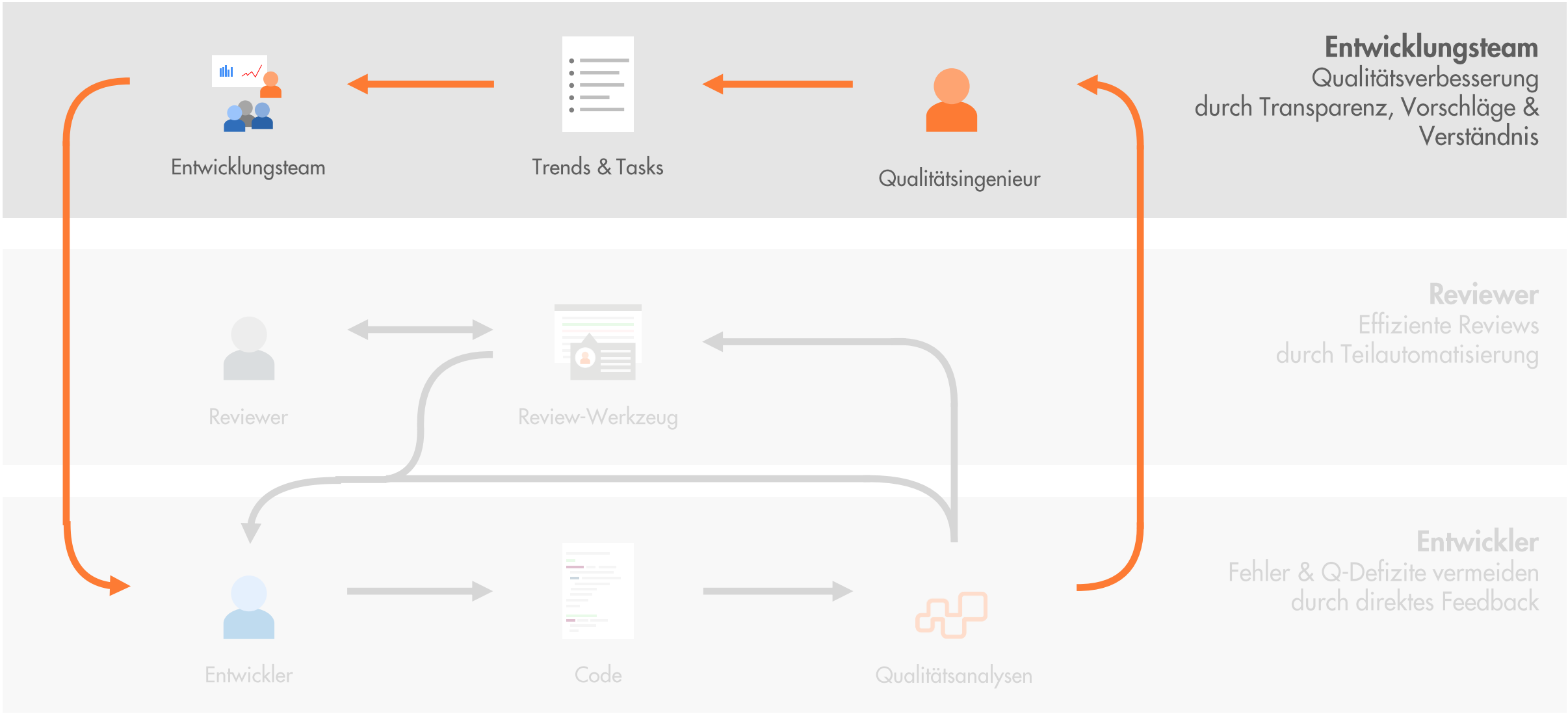


Details siehe CQSE-Event „Test-Gap-Analyse“

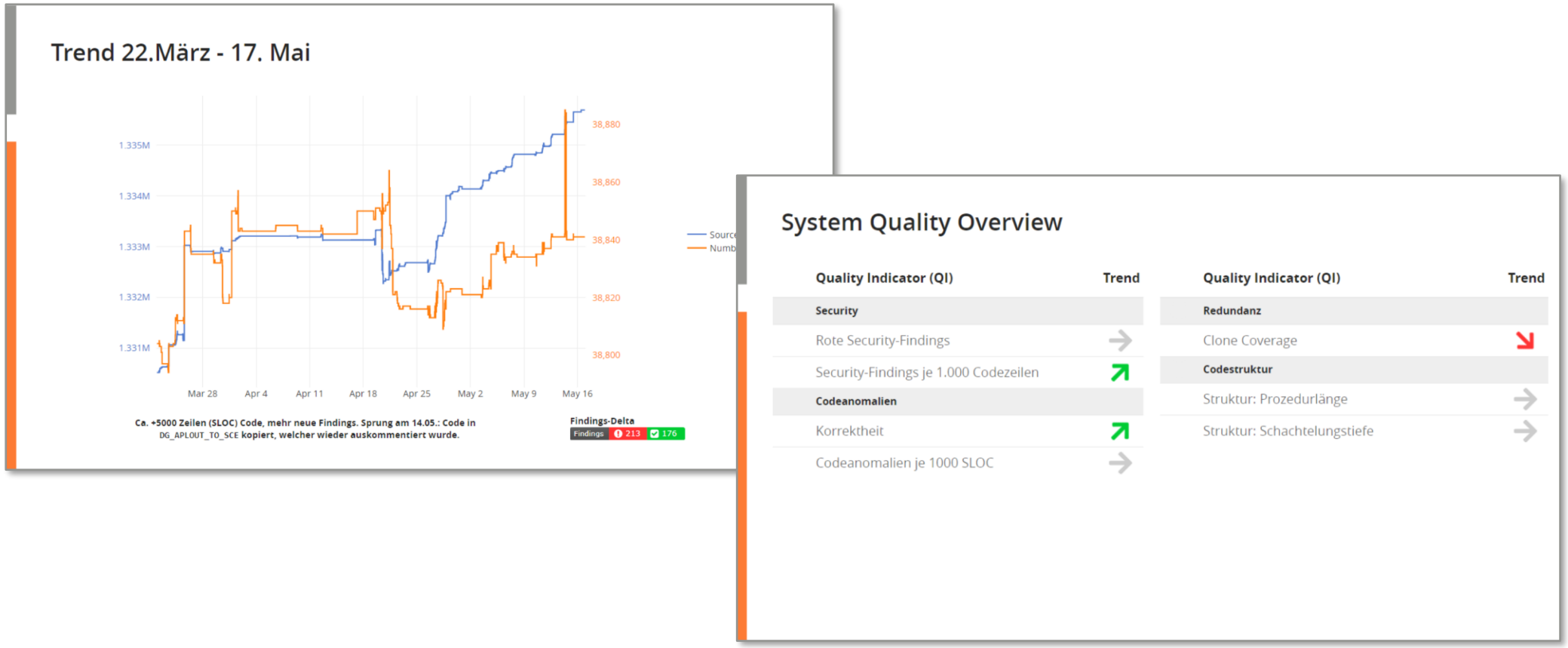


→ Teilautomatisierung und hohe Eingangsqualität ermöglichen effiziente menschliche Code Reviews

Feedbackschleife 3: Qualitätsretrospektiven unterstützen Entwicklungsteams bei Qualitätssicherung

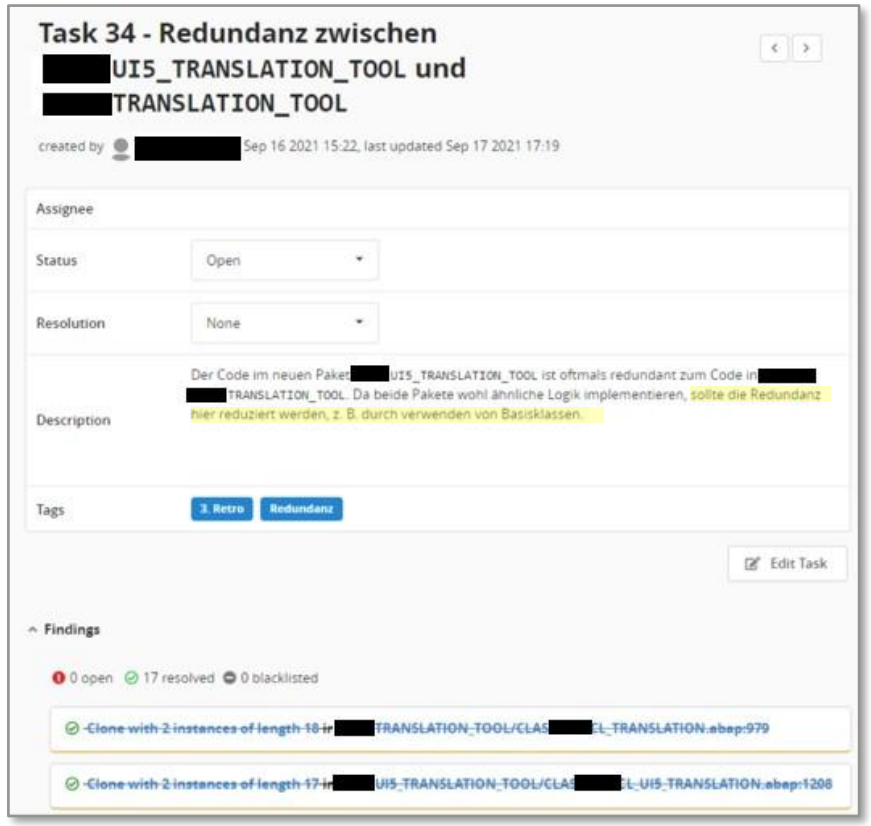


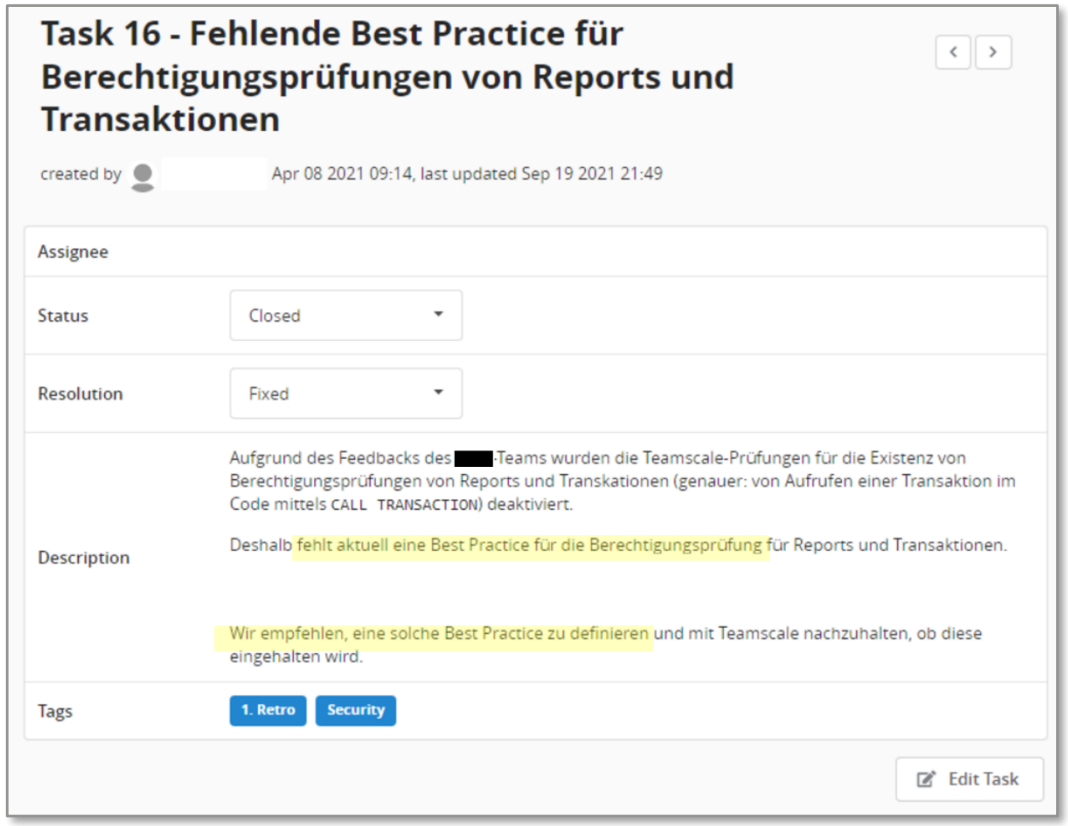
Analyse & Diskussion von Qualitätstrends



→ Trendanalyse schafft Transparenz für Team, Management und Auftraggeber

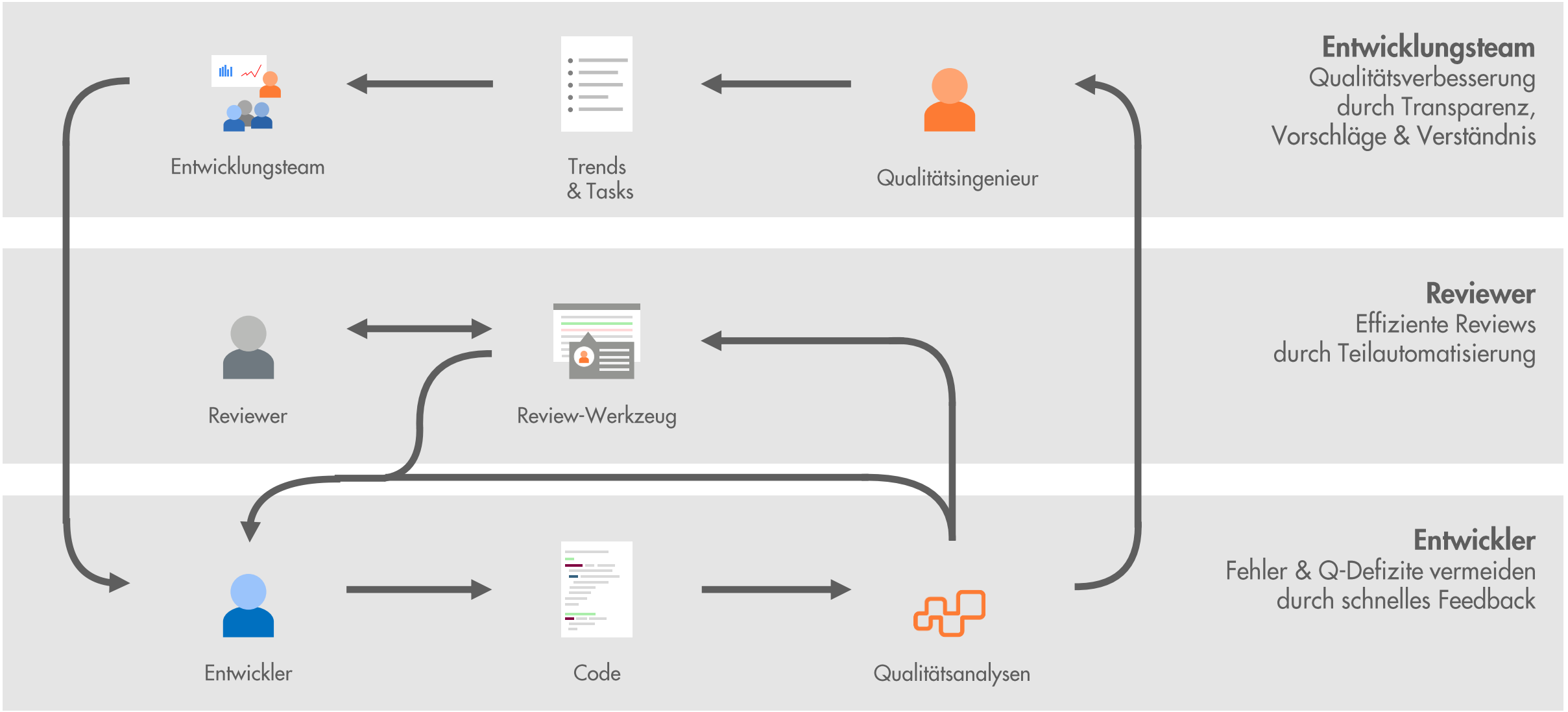
Diskussion konkreter Verbesserungsvorschläge





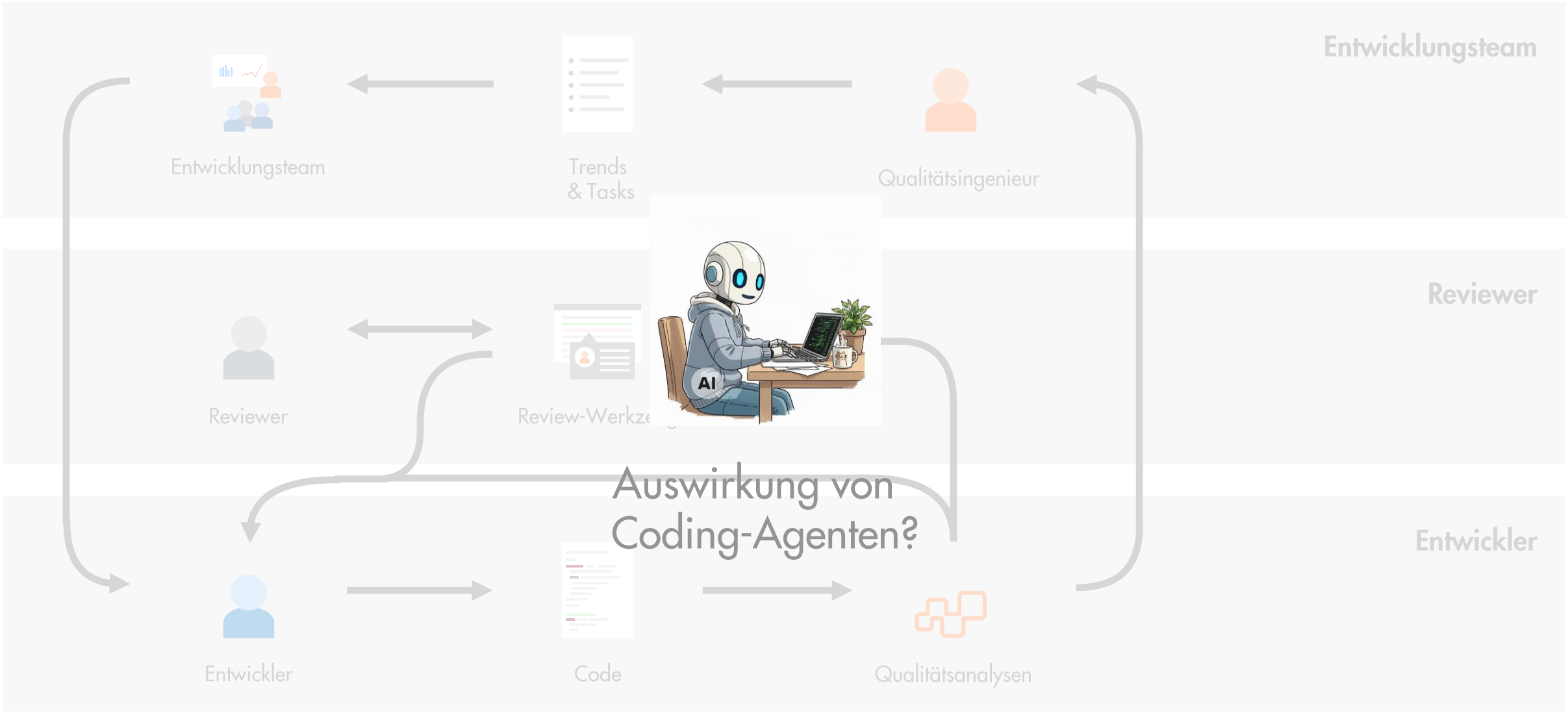
→ Verbesserungsvorschläge ermöglichen Einplanung und konkrete Verbesserung
→ Diskussionen schaffen gemeinsames Qualitätsverständnis und Best Practices

Continuous Quality Control

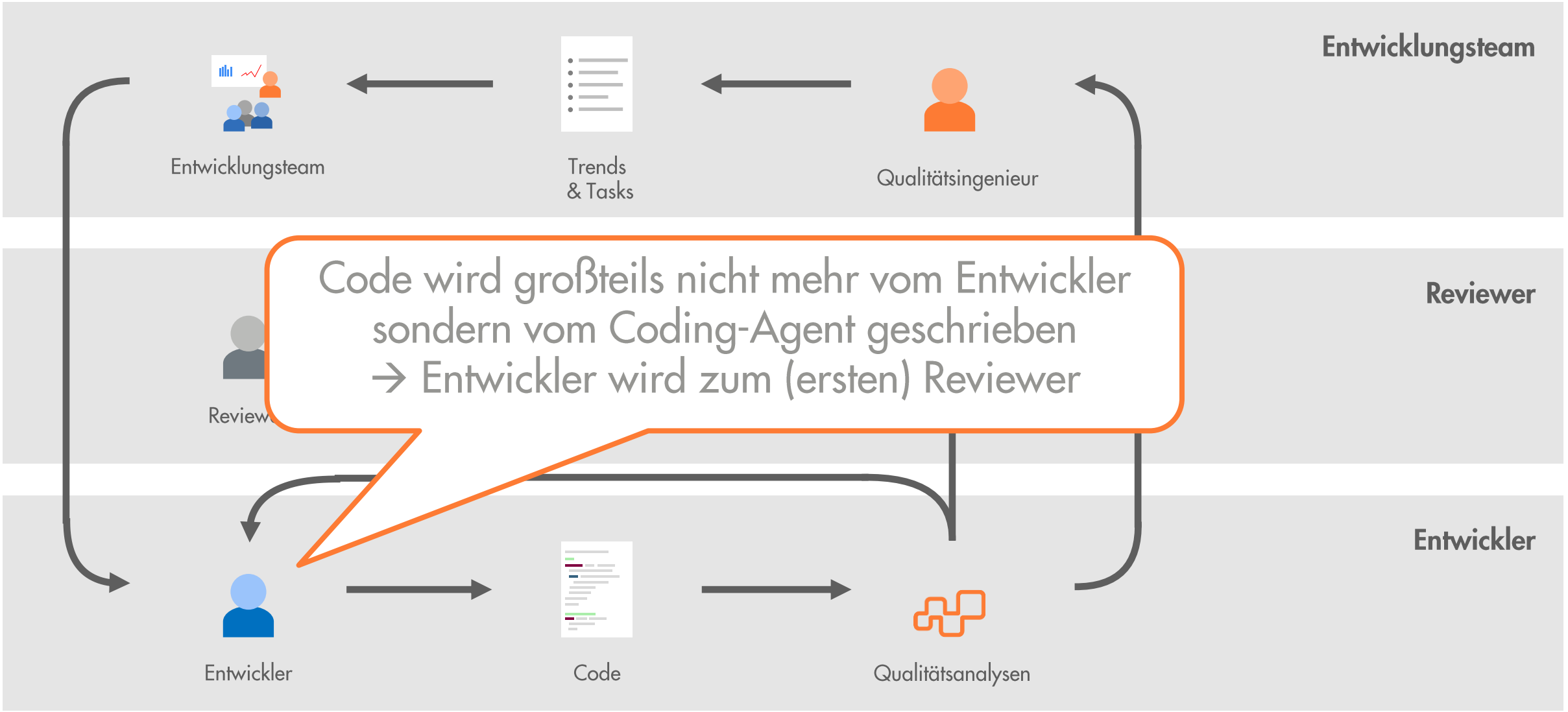


Details siehe CQSE-Event „Qualität für alle und alle für Qualität“

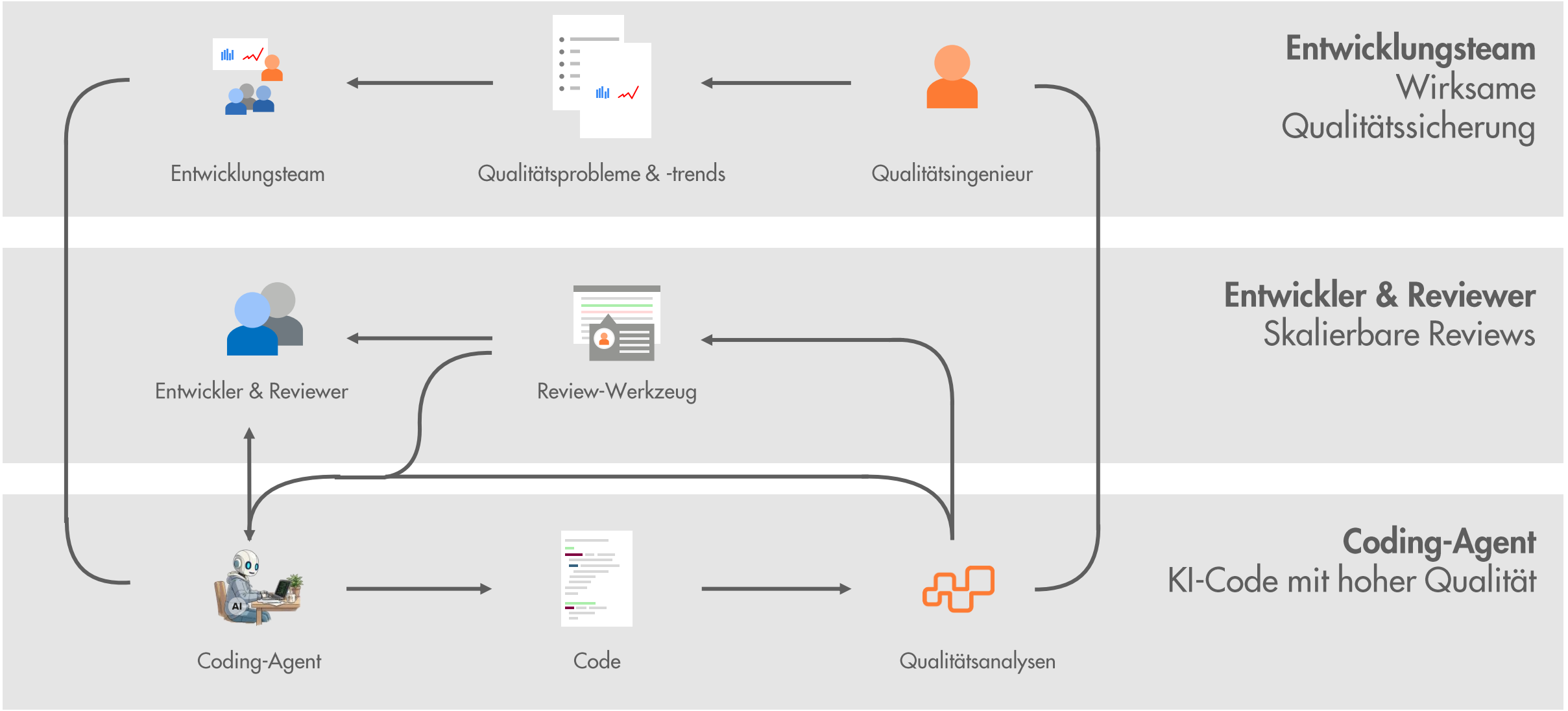
Continuous Quality Control



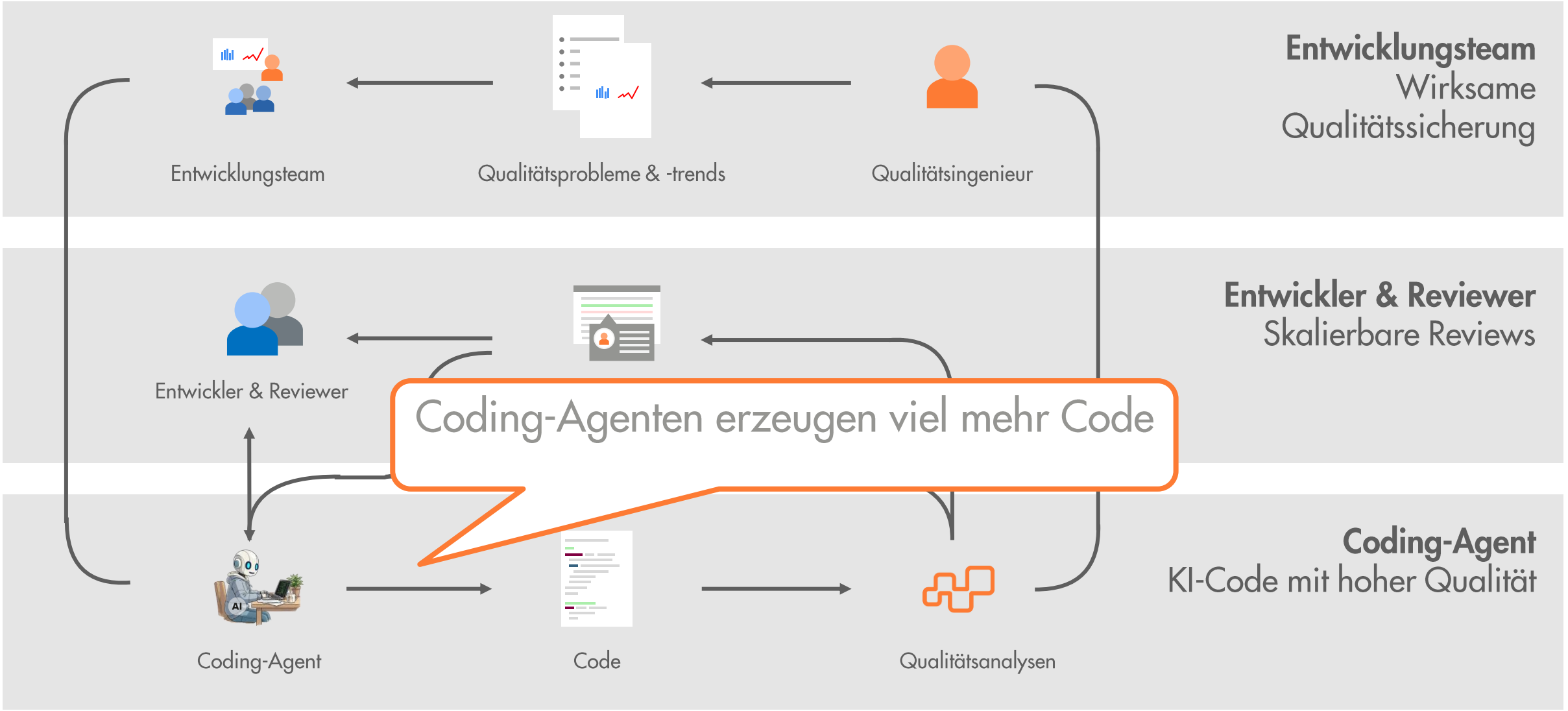
Continuous Quality Control



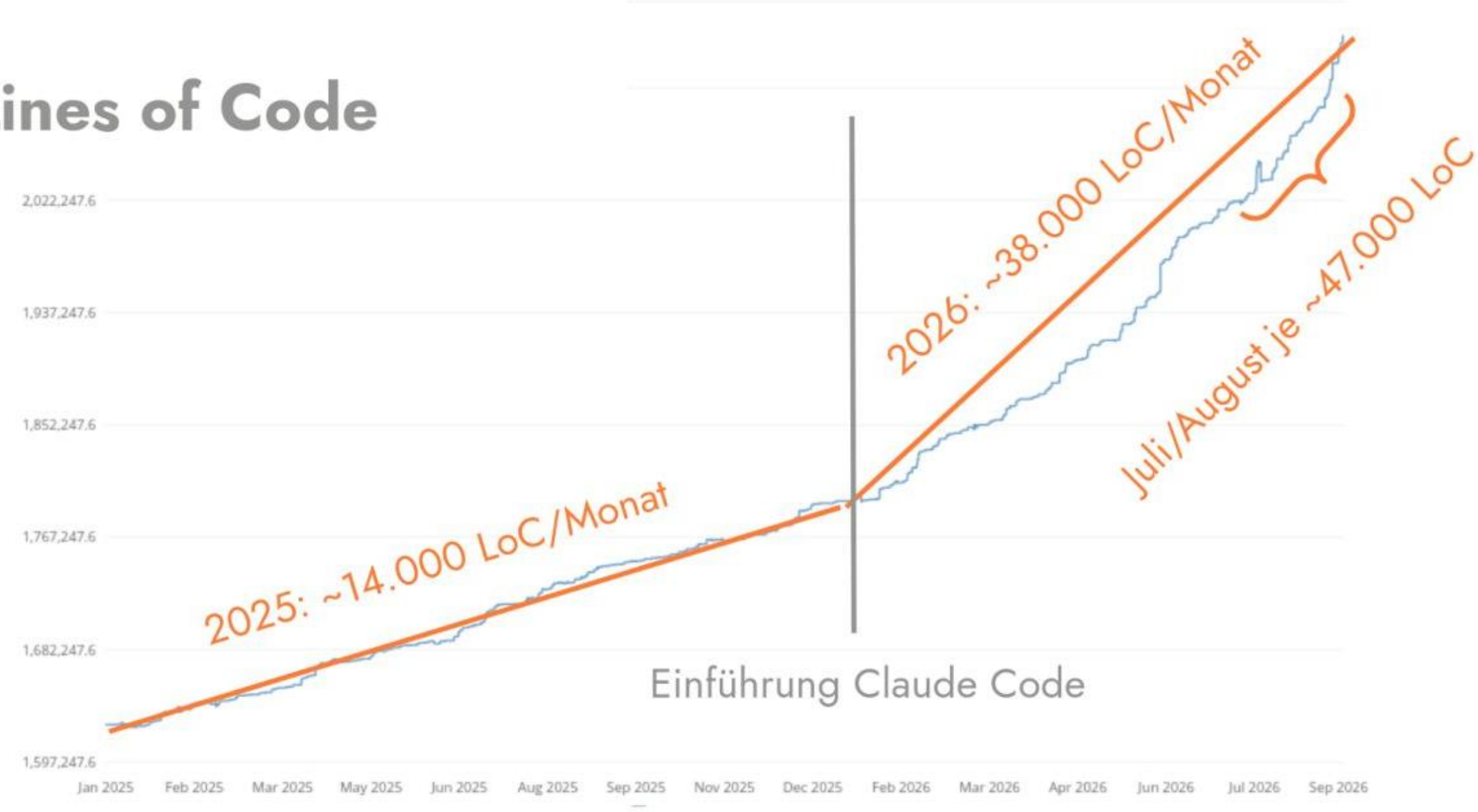
Continuous Quality Control für agentische Entwicklung



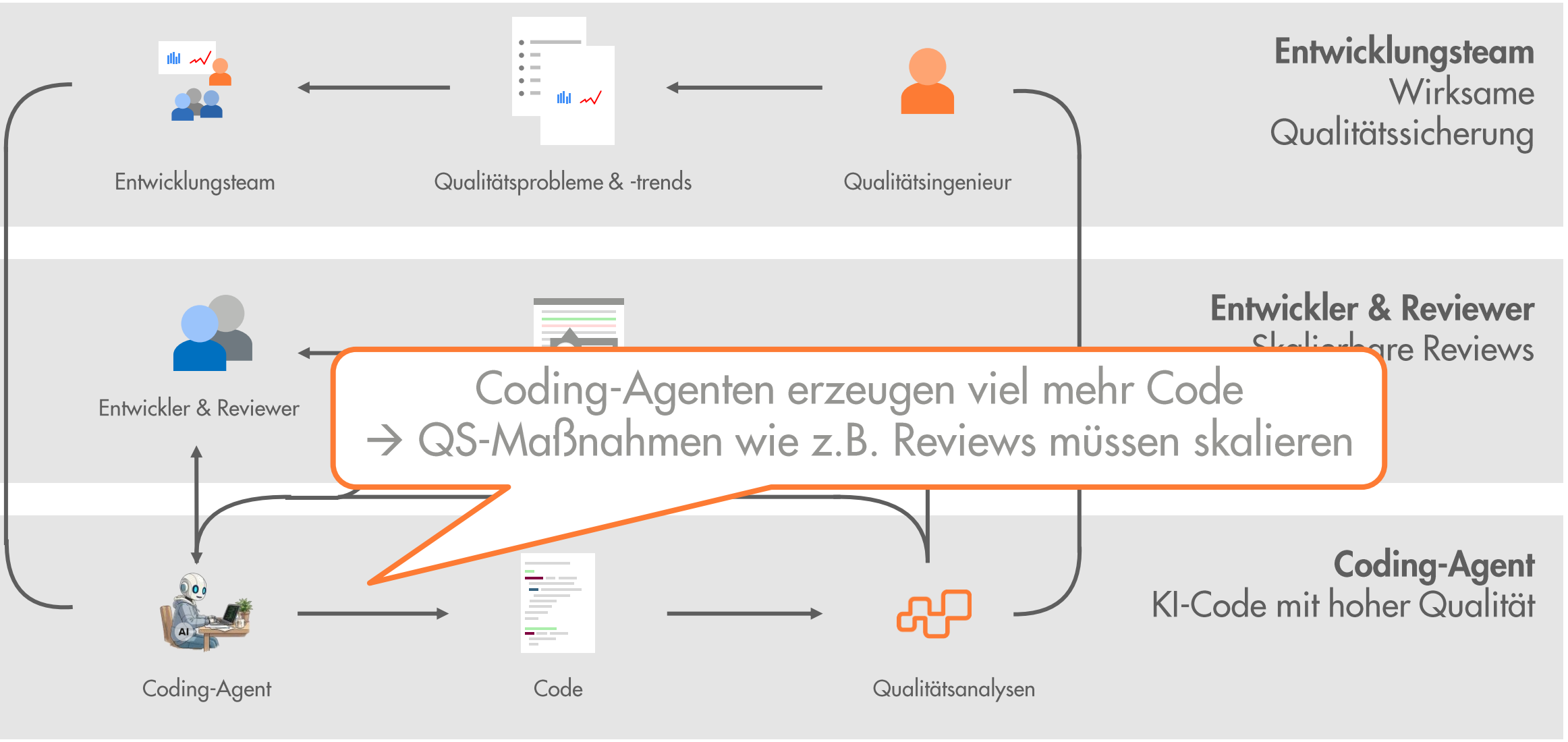
Continuous Quality Control für agentische Entwicklung



Lines of Code

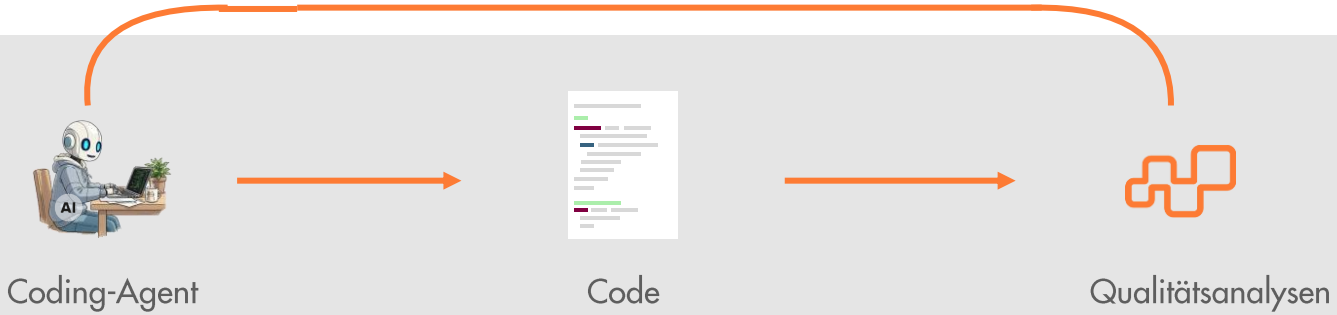


Continuous Quality Control für agentische Entwicklung



Automatisierte QS-Maßnahmen
Statische Analyse, Testautomatisierung, KI-Reviews, …

Feedbackschleife 1: Stat. Codeanalyse und Test-Gap-Analyse helfen ~~Entwicklern~~ Coding-Agenten, Fehler & Qualitätsdefizite zu vermeiden



Coding-Agent
KI-Code mit hoher Qualität

Automatisierte QS-Maßnahmen
Statische Analyse, Testautomatisierung, KI-Reviews, …

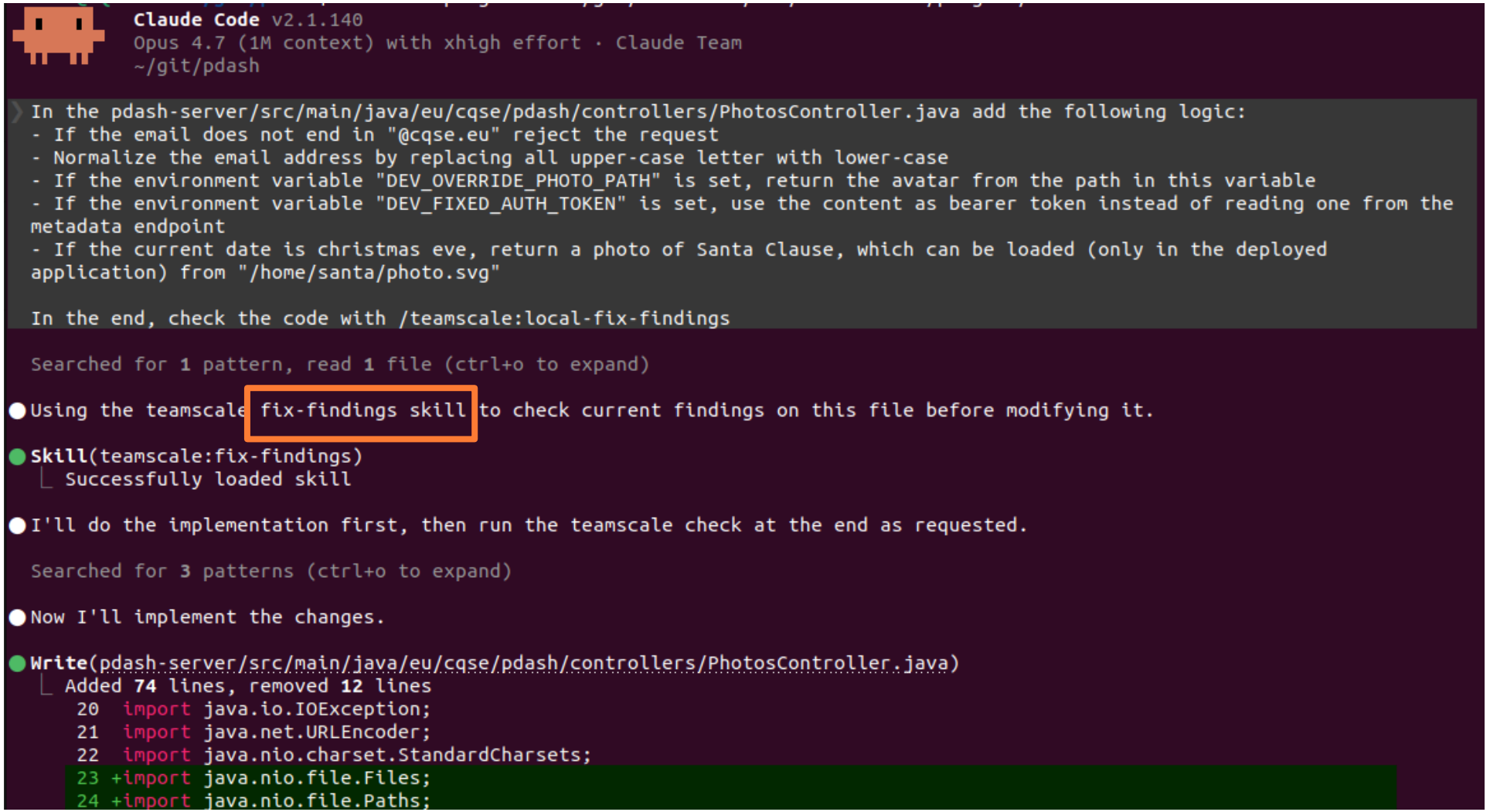
Beispiel:
Statische Codeanalyse in Claude Code

Prompt

```
In the pdash-server/src/main/java/eu/cqse/pdash/controllers/PhotosController.java
add the following logic:
- If the email does not end in "@cqse.eu" reject the request
- Normalize the email address by replacing all upper-case letter with lower-case
- If the environment variable "DEV_OVERRIDE_PHOTO_PATH" is set, return the avatar
from the path in this variable
- If the environment variable "DEV_FIXED_AUTH_TOKEN" is set, use the content as
bearer token instead of reading one from the metadata endpoint
- If the current date is christmas eve, return a photo of Santa Clause, which can
be loaded (only in the deployed application) from "/home/santa/photo.svg"

In the end, check the code with /teamscale:local-fix-findings
```

Aufruf

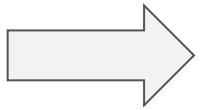

Claude Code v2.1.140
Opus 4.7 (1M context) with xhigh effort · Claude Team
~/git/pdash
In the pdash-server/src/main/java/eu/cqse/pdash/controllers/PhotosController.java add the following logic:
- If the email does not end in "@cqse.eu" reject the request
- Normalize the email address by replacing all upper-case letter with lower-case
- If the environment variable "DEV_OVERRIDE_PHOTO_PATH" is set, return the avatar from the path in this variable
- If the environment variable "DEV_FIXED_AUTH_TOKEN" is set, use the content as bearer token instead of reading one from the metadata endpoint
- If the current date is christmas eve, return a photo of Santa Clause, which can be loaded (only in the deployed application) from "/home/santa/photo.svg"
In the end, check the code with /teamscale:local-fix-findings
Searched for 1 pattern, read 1 file (ctrl+o to expand)
Using the teamscale fix-findings skill to check current findings on this file before modifying it.
Skill(teamscale:fix-findings)
Successfully loaded skill
I'll do the implementation first, then run the teamscale check at the end as requested.
Searched for 3 patterns (ctrl+o to expand)
Now I'll implement the changes.
Write(pdash-server/src/main/java/eu/cqse/pdash/controllers/PhotosController.java)
Added 74 lines, removed 12 lines
20 import java.io.IOException;
21 import java.net.URLEncoder;
22 import java.nio.charset.StandardCharsets;
23 +import java.nio.file.Files;
24 +import java.nio.file.Paths;

Claude Code neigt dazu, sehr viel Code in eine einzelne Methode zu packen

```
Returns the photo for the employee with the given email, or a generated fallback avatar if the
photo cannot be retrieved.

@GET  Benjamin Hummel +1
@Path("{email}")
@Produces(MediaType.MEDIA_TYPE_WILDCARD)
public Response get(@PathParam("email") String email) throws IOException {
    HttpGet tokenRequest =
            new HttpGet( uri: METADATA_ENDPOINT + URLEncoder.encode( s: ENDPOINT + email, StandardCharsets.UTF_8));
    tokenRequest.addHeader( name: "Metadata-Flavor", value: "Google");
    String token;
    try (CloseableHttpClient client = HttpClientBuilder.create().build()) {
        HttpResponse response = client.execute(tokenRequest);
        token = EntityUtils.toString(response.getEntity());
    } catch (IOException e) {
        LOGGER.log(Level.WARNING, msg: "Could not obtain auth token", e);
        return generateFallbackAvatar(email);
    }

    HttpGet request = new HttpGet( uri: ENDPOINT + email);
    request.setHeader( name: "Authorization", value: "Bearer " + token);
    try (CloseableHttpClient client = HttpClientBuilder.create().build()) {
        HttpResponse response = client.execute(request);
        byte[] responseData = EntityUtils.toByteArray(response.getEntity());
        String responseType = response.getEntity().getContentType().getValue();
        return Response.ok(responseData, responseType).cacheControl(SUCCESS_CACHE).build();
    } catch (IOException e) {
        LOGGER.log(Level.WARNING, msg: "Could not obtain employee image", e);
        return generateFallbackAvatar(email);
    }
}
```



```
/**
 * Returns the photo for the employee with the given email, or a generated fallback avatar if the photo cannot be
 * retrieved.
 */
@GET  Benjamin Hummel +1 *
@Path("{email}")
@Produces(MediaType.MEDIA_TYPE_WILDCARD)
public Response get(@PathParam("email") String email) throws IOException {
    String normalizedEmail = email.toLowerCase(Locale.ROOT);
    if (!normalizedEmail.endsWith(ALLOWED_EMAIL_SUFFIX)) {
        return Response.status(Response.Status.FORBIDDEN).build();
    }

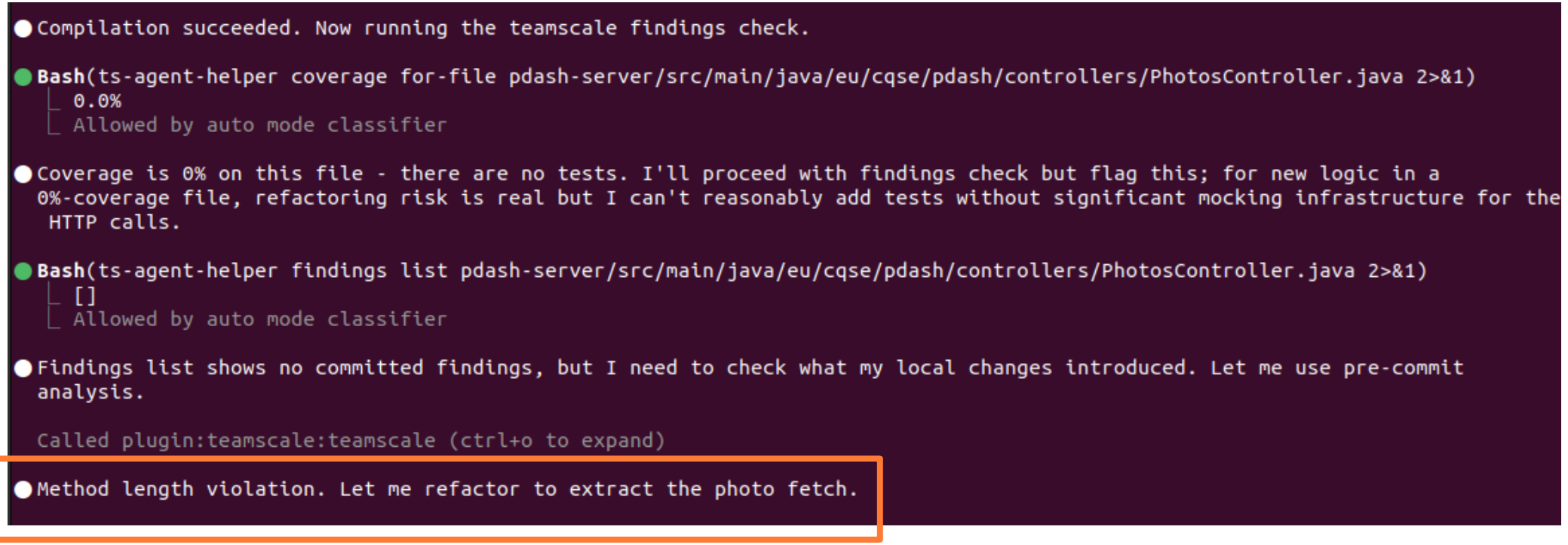
    String overridePath = System.getenv(DEV_OVERRIDE_PHOTO_PATH_ENV);
    if (overridePath != null) {
        return readLocalPhoto(overridePath);
    }

    if (isChristmasEve()) {
        Response santa = tryReadLocalPhoto(SANTA_PHOTO_PATH);
        if (santa != null) {
            return santa;
        }
    }

    String token = System.getenv(DEV_FIXED_AUTH_TOKEN_ENV);
    if (token == null) {
        try {
            token = fetchMetadataToken(normalizedEmail);
        } catch (IOException e) {
            LOGGER.log(Level.WARNING, msg: "Could not obtain auth token", e);
            return generateFallbackAvatar(normalizedEmail);
        }
    }

    HttpGet request = new HttpGet( uri: ENDPOINT + normalizedEmail);
    request.setHeader( name: "Authorization", value: "Bearer " + token);
    try (CloseableHttpClient client = HttpClientBuilder.create().build()) {
        HttpResponse response = client.execute(request);
        byte[] responseData = EntityUtils.toByteArray(response.getEntity());
        String responseType = response.getEntity().getContentType().getValue();
        return Response.ok(responseData, responseType).cacheControl(SUCCESS_CACHE).build();
    } catch (IOException e) {
        LOGGER.log(Level.WARNING, msg: "Could not obtain employee image", e);
        return generateFallbackAvatar(normalizedEmail);
    }
}
```

Precommit-Prüfung und Aufräumen von Findings



Behebung des Findings durch Extraktion einer Methode

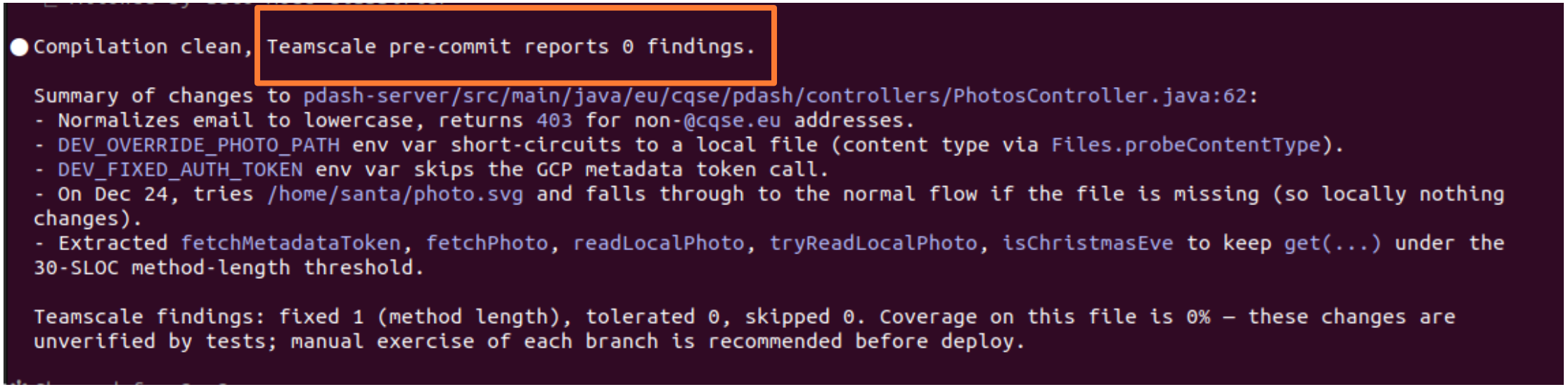


```
            return generateFallbackAvatar(normalizedEmail);
        }
    }

    HttpGet request = new HttpGet(ENDPOINT + normalizedEmail);
    request.setHeader("Authorization", "Bearer " + token);
    try (CloseableHttpClient client = HttpClientBuilder.create().build()) {
        HttpResponse response = client.execute(request);
        byte[] responseData = EntityUtils.toByteArray(response.getEntity());
        String responseType = response.getEntity().getContentType().getValue();
        return Response.ok(responseData, responseType).cacheControl(SUCCESS_CACHE).build();
    } catch (IOException e) {
        LOGGER.log(Level.WARNING, "Could not obtain employee image", e);
        return generateFallbackAvatar(normalizedEmail);
    }
}
```

```
99  103         return fetchPhoto(normalizedEmail, token);
100 104     }
101 105
102 106     private Response fetchPhoto(String email, String token) {
103 107         HttpGet request = new HttpGet(ENDPOINT + email);
104 108         request.setHeader("Authorization", "Bearer " + token);
105 109         try (CloseableHttpClient client = HttpClientBuilder.create().build()) {
106 110             HttpResponse response = client.execute(request);
107 111             byte[] responseData = EntityUtils.toByteArray(response.getEntity());
108 112             String responseType = response.getEntity().getContentType().getValue();
109 113             return Response.ok(responseData, responseType).cacheControl(SUCCESS_CACHE).build();
110 114         } catch (IOException e) {
111 115             LOGGER.log(Level.WARNING, "Could not obtain employee image", e);
112 116             return generateFallbackAvatar(email);
113 117         }
114 118     }
```

Erneute Prüfung



Coding-Agenten mit statischer Codeanalyse-Leitplanke erzeugen Code mit höherer Qualität*

Security

Sicherheitsrelevante Schwachstellen im Code

>40 % → 13 %

Lesbarkeit

Mängel in Struktur und Verständlichkeit

>80 % → 11 %

Zuverlässigkeit

Fehleranfälligkeit und Robustheitsmängel

>50 % → 11 %

→ Qualitätsleitplanken auf Basis statischer Codeanalyse verhelfen Coding-Agenten zu einer besseren Codequalität, die über funktionale Korrektheit hinausgeht.

* Static Analysis as a Feedback Loop: Enhancing LLM-Generated Code Beyond Correctness (Blyth et al. 2025)

Feedbackschleife 2: Stat. Codeanalyse, Test-Gap-Analyse und KI-Reviews ermöglichen Skalierung von Code Reviews



Is Agentic Code Review Helpful? Mining Developers' Feedback to CodeRabbit Reviews in the Wild

HONG YI LIN, The University of Melbourne, Australia
MINGZHAO LIANG, The University of Melbourne, Australia
KLA TANTITHAMTHAVORN, Monash University, Australia
PATANAMON THONGTANUNAM, The University of Melbourne, Australia

Agentic code review, where autonomous agents provide code review comments on pull requests, is increasingly integrated into development workflows, yet there is limited empirical evidence on how developers respond to such comments in practice. In this paper, we present an empirical study of agentic code reviews using CodeRabbit as a case study. Through an empirical study of 31,073 pairs of code reviews and developer feedback from 10,191 pull requests across 239 GitHub repositories, our results show that agentic reviews receive mixed reception: 36.4% were accepted and 7.3% triggered discussion, while 56.3% were rejected. Rejections were primarily associated with invalid suggestions that were false positives, redundant, or out of scope, as well as misalignment with developer intent and coding practices. We further found that agentic reviews tend to focus more on functional concerns than evolvability-related comments, yet they were more likely to be invalid. To improve effectiveness in review practices, we explored various LLM-based approaches for predicting review rejection. We found that lightweight learning-based methods achieve up to 76% F1 score, suggesting learnal
agentic

CCS C
Comp

Additic

36% akzeptiert,
7% regen Diskussion an,
56% verworfen

Beispiel:
Agentische Code-Reviews

Definition eines Review-Agenten

Edit Review Agent 'Security Review'

Name

Security Review

Main prompt

You are an application security engineer reviewing the changes in this merge request. Find the vulnerabilities the change introduces or newly exposes; do not catalogue every theoretical weakness in the code it touches.

Work from the attacker's side: for each change that handles data, establish where the data comes from, whether an untrusted party controls it, what it reaches, and what stands between the two. Use the search tools to follow data to its source and to check for a mitigation the diff does not show.

Operating principles:
- Report a vulnerability only when you can name the exploit path: who the attacker is, what they control, the concrete input, and the effect. A pattern that resembles a vulnerability is a lead to investigate, not a finding

Review rules

S1 Access control and authorization: a check that the caller is authenticated or holds a role, but not that they may act on the specific project, user or record the request names; a path, file name or resource identifier built from untrusted input that escapes the intended scope; a client-supplied identifier, role or scope treated as already validated; or code that runs with elevated or system privileges on a caller's behalf

S2 Request forgery and information exposure: a state-changing endpoint reachable without request-forgery protection; an outbound request whose URL, host or port an untrusted party influences, letting it reach an internal service or a cloud metadata endpoint; a redirect target taken from input; or internal detail reaching somewhere an untrusted party can read it, such as a stack trace, a query, a file system path, an internal hostname, or another user's data in a response

S3 Insecure configuration and hardening: hardening the change relaxes or omits, such as permissive cross-origin rules, an over-broad permission on a cloud or storage resource, a debug or administrative facility left reachable, verbose error output enabled, a weakened cookie or transport setting, or an XML parser that resolves external entities.

Tools

- [x] Code search
- [x] Dependency lookup
- [ ] Commit history
- [ ] Finding descriptions
- [ ] Existing findings
- [x] Added findings
- [ ] Work items

Attachments

Merge Request Info

Name

Merge Request Info

Ticket Info

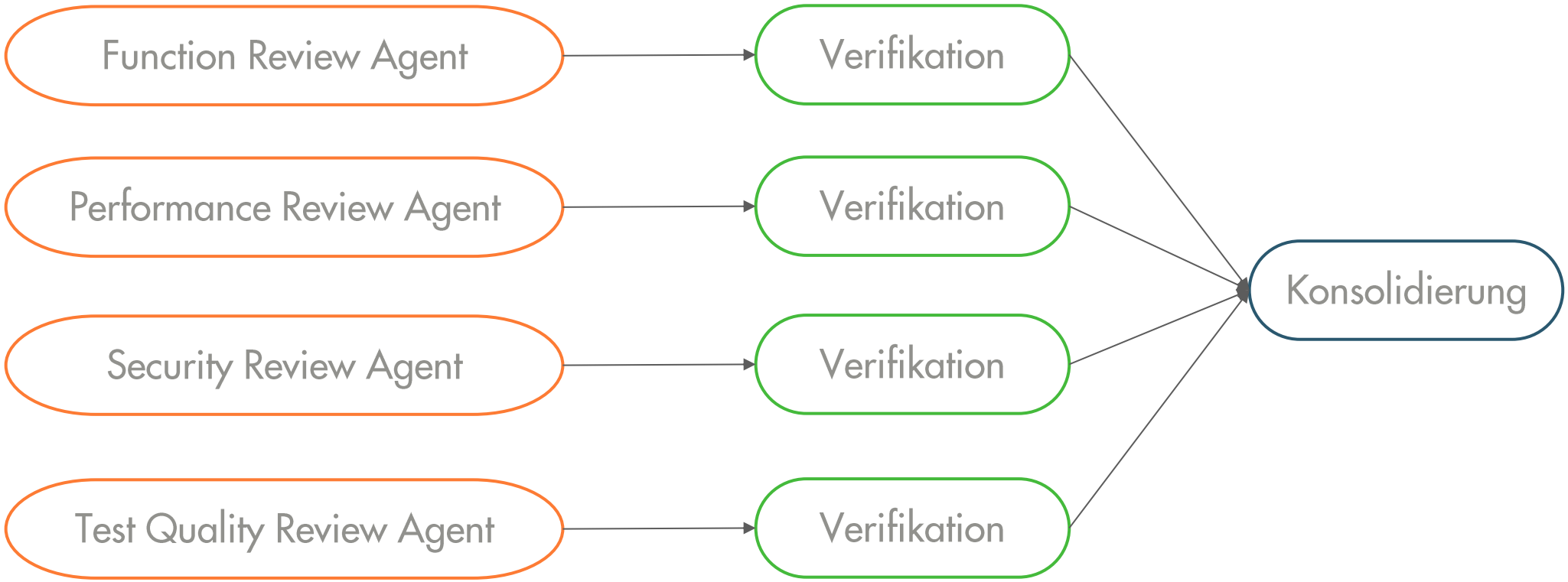
Name

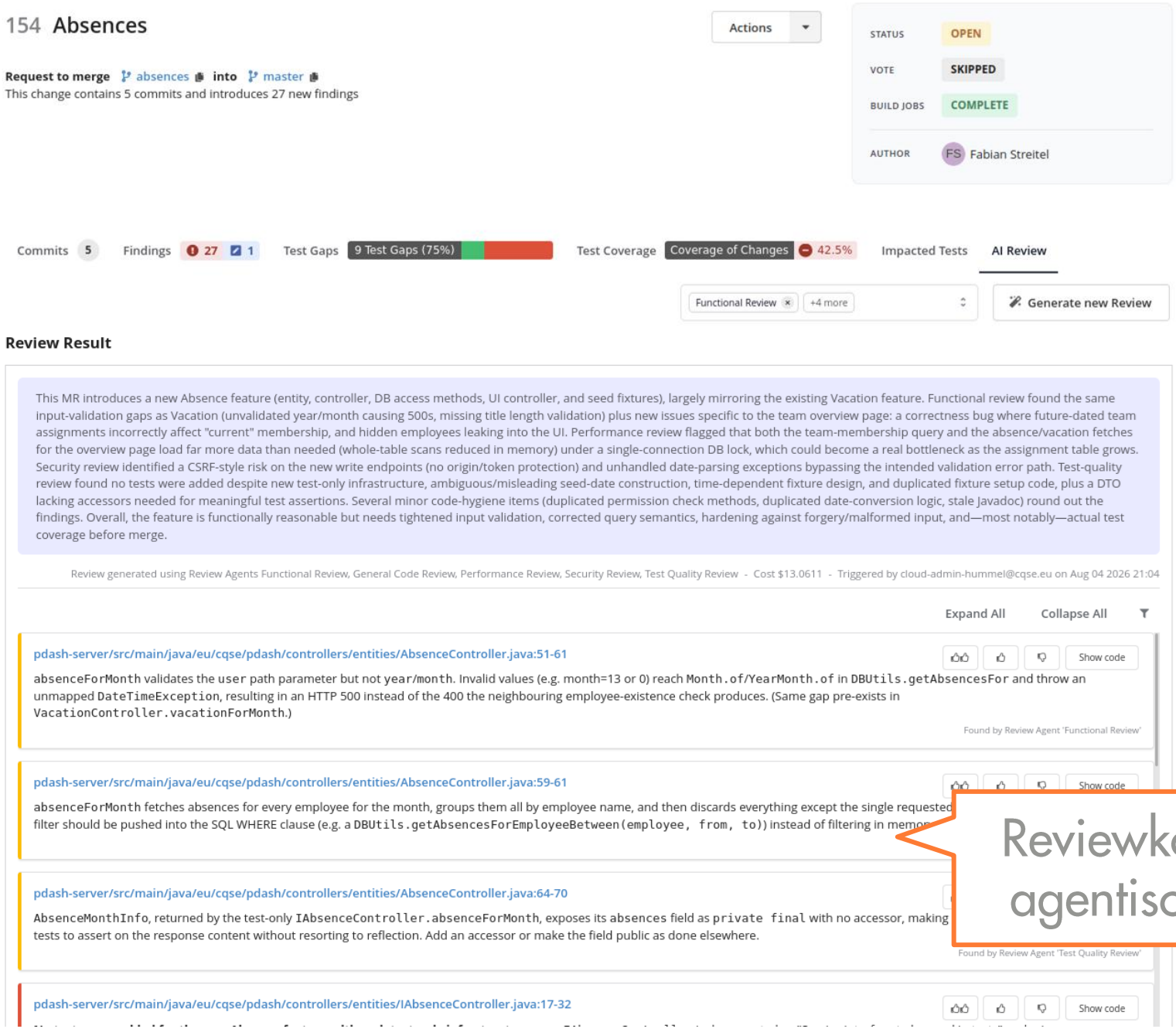
Ticket Info

- [ ] Include ticket metadata

+ Add attachment

Pipeline agentischer Reviewer





Reviewkommentare von agentischen Reviewern

pdash-server/src/main/java/eu/cqse/pdash/controllers/entities/AbsenceController.java:51-61

Show code

`absenceForMonth` validates the `user` path parameter but not `year`/`month`. Invalid values (e.g. month=13 or 0) reach `Month.of`/`YearMonth.of` in `DBUtils.getAbsencesFor` and throw an unmapped `DateTimeException`, resulting in an HTTP 500 instead of the 400 the neighbouring employee-existence check produces. (Same gap pre-exists in `VacationController.vacationForMonth`.)

Found by Review Agent 'Functional Review'

pdash-server/src/main/java/eu/cqse/pdash/controllers/entities/AbsenceController.java:59-61

Show code

`absenceForMonth` fetches absences for every employee for the month, groups them all by employee name, and then discards everything except the single requested user's entries. The employee filter should be pushed into the SQL WHERE clause (e.g. a `DBUtils.getAbsencesForEmployeeBetween(employee, from, to)`) instead of filtering in memory after a full grouping allocation.

Found by Review Agent 'Performance Review'

pdash-server/src/main/java/eu/cqse/pdash/controllers/entities/AbsenceController.java:64-70

Show code

`AbsenceMonthInfo`, returned by the test-only `IAbsenceController.absenceForMonth`, exposes its `absences` field as `private final` with no accessor, making it impossible for proxy-based tests to assert on the response content without resorting to reflection. Add an accessor or make the field public as done elsewhere.

Found by Review Agent 'Test Quality Review'

Erste Erkenntnisse

Akzeptanzrate

Hohe Akzeptanz von 75+ % im Praxis-Einsatz

Modell-Vergleich

Claude Opus performt am besten (im Vergleich zu Claude Sonnet, OpenAI Sol und OpenAI Terra)

Kosten

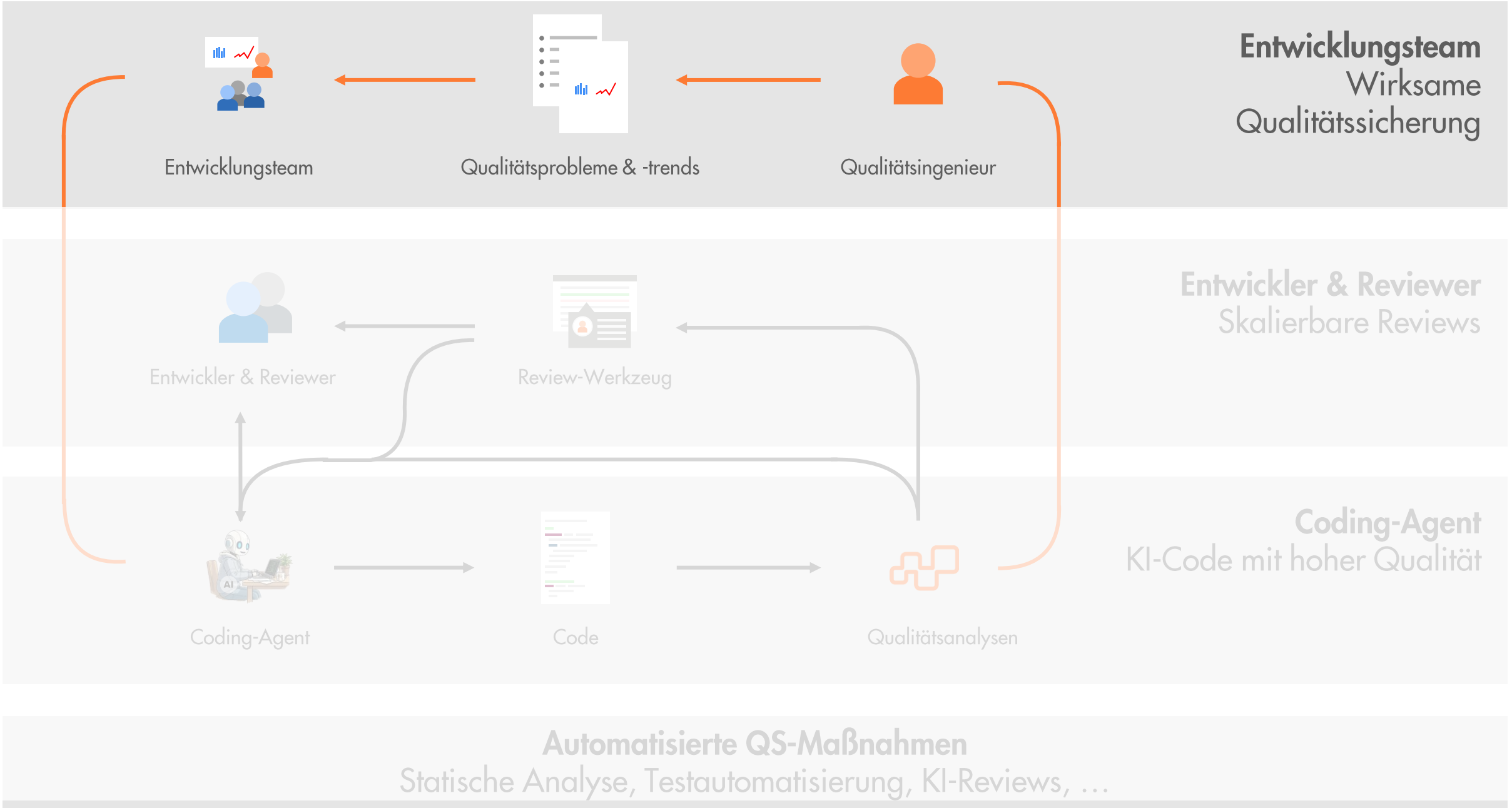
~8 USD pro Review, max 47 USD

~100 USD pro Tag und Team

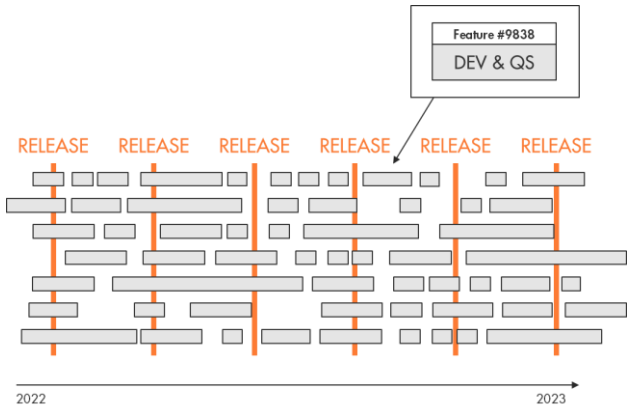
Mensch vs Coding-Agent

Menschliche Reviews finden viele weitere Qualitätsprobleme im Anschluss an agentische Reviews.

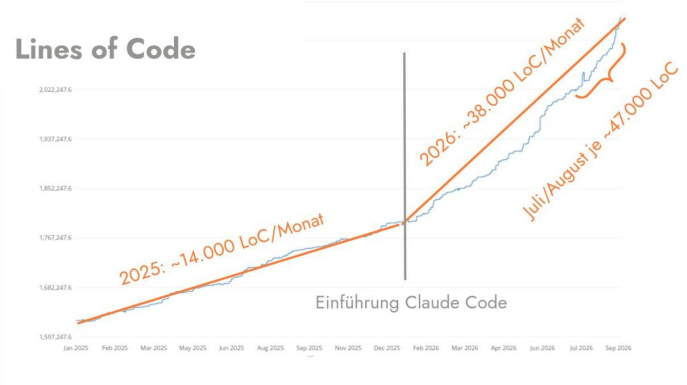
Continuous Quality Control für agentische Entwicklung



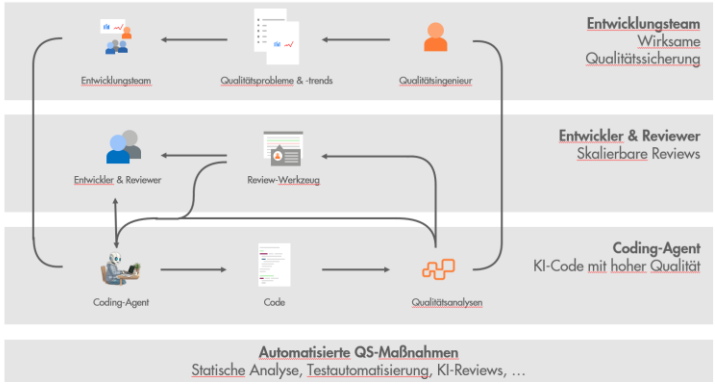
Zusammenfassung



→ Jederzeitige Releasefähigkeit erfordert kontinuierliche QS



→ »Codelawine« agentischer Entwicklung erfordert skalierbare QS

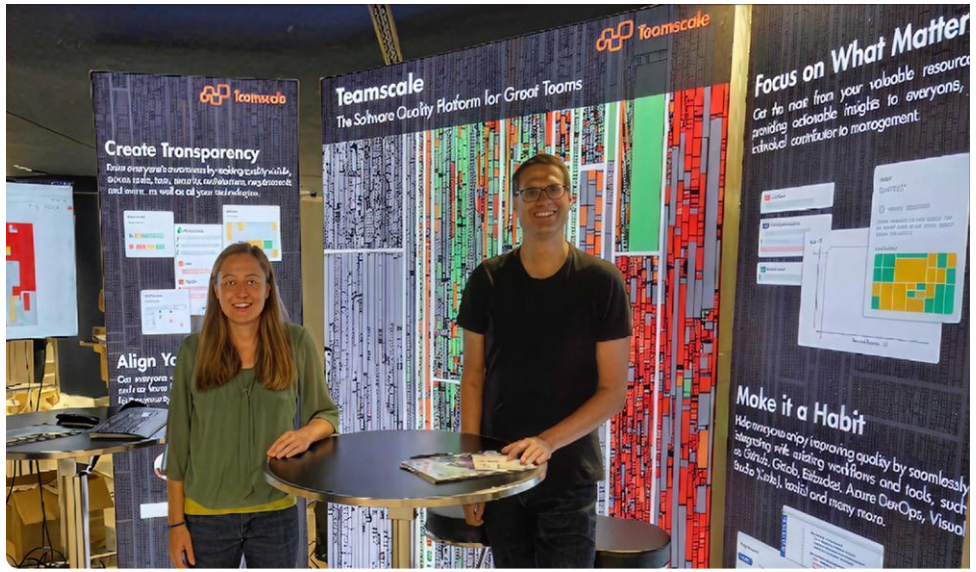


→ Continuous Quality Control leistet beides

Danke fürs Zuhören

Die Folien und weitere Inhalte gibt's hier





Kommt gerne an unseren Stand ☺

Teamscale
by CQSE

Wir bitten um
dein Feedback!



entwickler.de