

Codequalität vs KI

Chancen, Grenzen & Erfahrungen mit GenAI in der Qualitätssicherung

BASTA! Herbst 2026



Dr. Tobias Röhm



Dr. Tobias Röhms



Freelance-
Entwickler



CQSE



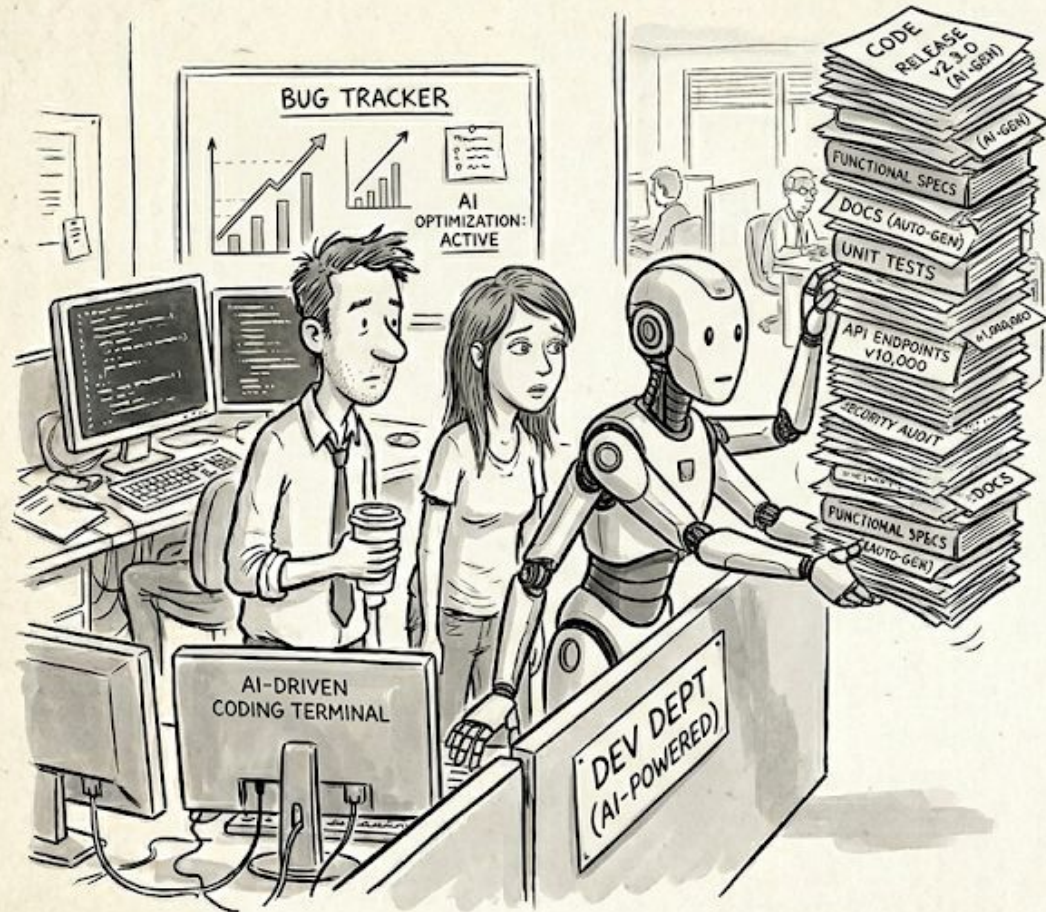
2004 - 2010

2010 - 2015

2015 - Heute







Heaven

»Coding-Agenten werden breit eingesetzt«

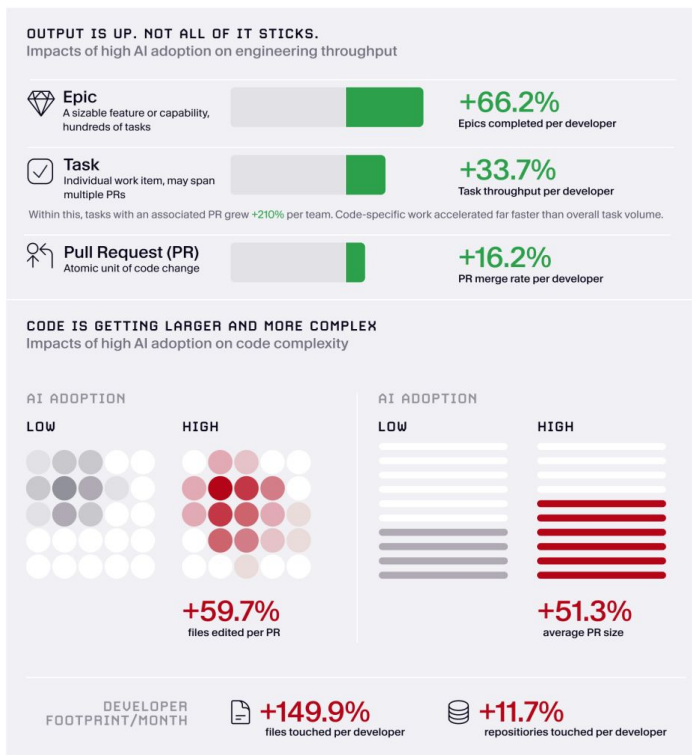


»68 % der professionellen Entwickler setzen Coding-Agenten täglich ein,
90 % der professionellen Entwickler setzen Coding-Agenten wöchentlich ein.« [1]



»47% des Codes sind komplett von Coding-Agenten geschrieben,
38 % des Codes sind mit Hilfe von Coding-Agenten geschrieben.« [1]

»Durch Coding-Agenten wird viel mehr Code erzeugt«



»Qualität von KI-Code ist manchmal schlecht«



Incidents pro Pull Request: +243%
Bugs pro Entwickler: +54 %



“Software Delivery Instability” hat zugenommen

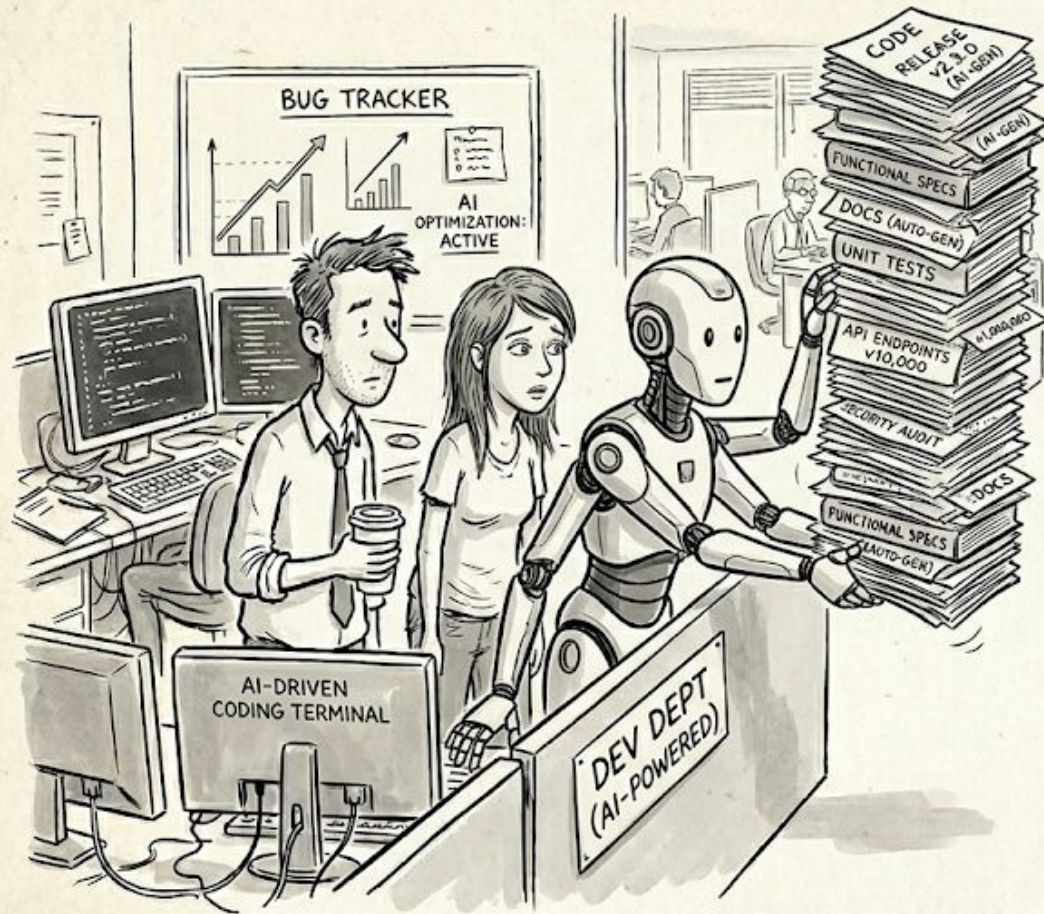


»Qualität von KI-Code ist wichtig«

Hohe Qualitätsanforderungen für missionskritische Software:
Funktionale Korrektheit, Freiheit von Bugs & Sicherheitslücken,
Wartbarkeit, ...

Qualität verschlechtert sich durch Agenten-Adoption
(ohne gezielte Gegenmaßnahmen)

Agenten-Geschwindigkeit und -Qualität korreliert mit Codequalität



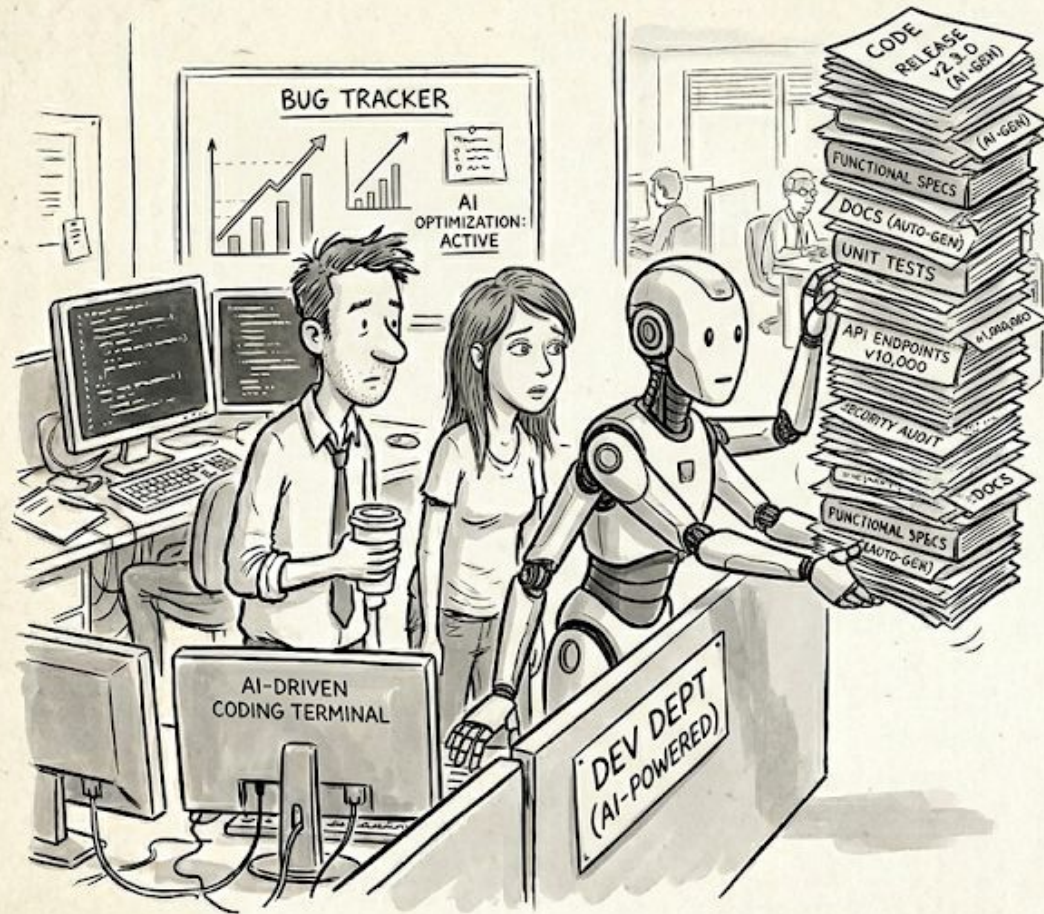
Wie skalieren Code-Reviews?



Wie bringt man Coding-Agenten dazu, hohe Qualität zu liefern?

Wie sieht ein QS-Prozess für agentische Entwicklung aus?

Heise



Wie skalieren Code-Reviews?



Wie bringt man Coding-Agenten dazu, hohe Qualität zu liefern?

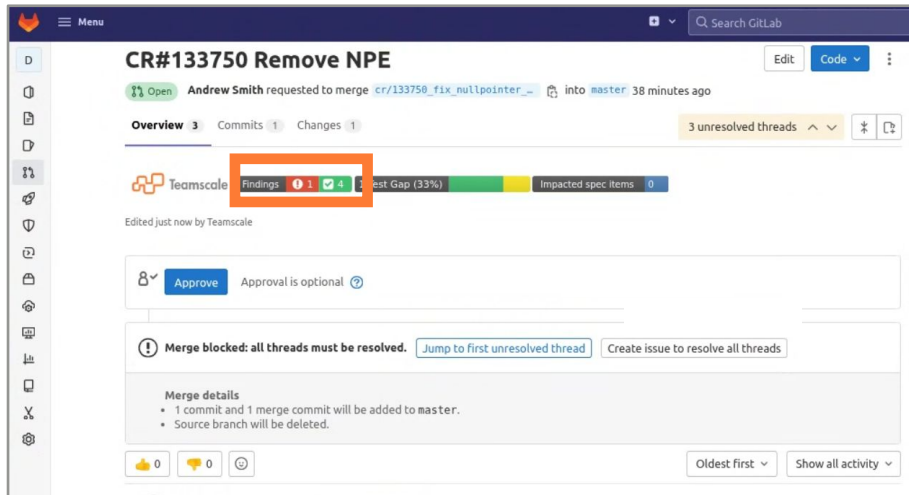
Wie sieht ein QS-Prozess für agentische Entwicklung aus?

Heaven



Automatisierung von Code-Reviews durch statische Analyse und Testautomatisierung

Annotation von Findings aus statischer Analyse an Pull Requests



The screenshot displays a GitLab pull request for the title "CR#133750 Remove NPE". The pull request was requested by Andrew Smith to merge the branch `cr/133750_fix_nullpointer_` into the `master` branch, 38 minutes ago. The interface shows the "Overview" tab with 3 commits and 1 change. A yellow banner indicates "3 unresolved threads". A Teamscale integration shows a "Findings" section with 1 error (red circle with exclamation mark) and 4 warnings (yellow circle with exclamation mark). A progress bar shows a "Test Gap (33%)" and "Impacted spec items: 0". Below this, there is an "Approve" button and a note that "Approval is optional". A message states "Merge blocked: all threads must be resolved." with links to "Jump to first unresolved thread" and "Create issue to resolve all threads". The "Merge details" section indicates that 1 commit and 1 merge commit will be added to master, and the source branch will be deleted. At the bottom, there are reaction buttons (thumbs up, thumbs down, smiley) and a "Show all activity" link.

CR#133750 Remove NPE

Andrew Smith requested to merge `cr/133750_fix_nullpointer_` into `master` 38 minutes ago

Overview 3 Commits 1 Changes 1 3 unresolved threads

Teamscale Findings 1 4 Test Gap (33%) Impacted spec items 0

Edited just now by Teamscale

Approve Approval is optional

! Merge blocked: all threads must be resolved. [Jump to first unresolved thread](#) [Create issue to resolve all threads](#)

Merge details

- 1 commit and 1 merge commit will be added to master.
- Source branch will be deleted.

0 0 0 Oldest first Show all activity

Annotation von Test Gaps an Pull Requests

CR#133750 Remove NPE

Open Andrew Smith requested to merge `cr/133750_fix_nullpointer_...` into `master` 38 minutes ago

Overview 3 Commits 1 Changes 1 3 unresolved threads

Teamscale Findings 1 1 4 1 Test Gap (33%) impacted spec 0

Edited just now by Teamscale

Approve Approval is optional

Merge blocked: all threads must be resolved. [Jump to first unresolved thread](#) [Create issue to resolve all threads](#)

Merge details

- 1 commit and 1 merge commit will be added to master.
- Source branch will be deleted.

0 0

Oldest first Show all activity

6 CR#133750 Remove NPE [Delta](#) [View on GitLab](#)

Request to merge `cr/133750_fix_nullpointer_bug` into `master` **OPEN** **COMMENTED**

This change contains 2 commits and introduces 1 new finding

Relevant Build Jobs 0

Test Gaps 1 Test Gap (33%)

Jan 12 2023 20:12 - Now | Test Gap: 33.3% Coverage sources All

[Search](#) [Help](#) [Feedback](#)

`bank/src/main/java/io/digisic/bank/service/AccountService.java`

Method	debitTransaction
State	Method existed at the baseline and was not tested after its latest modification
Last changed	Today 06:54

Automatisierung von Code-Reviews durch KI-Reviews

Ergebnis nach der Einführung eines führenden “AI Review Tools” bei einem großen Konzern (mehrere 1000 Entwickler:innen) begleitet durch den Tool-Hersteller über einen Zeitraum von über einem Jahr:

Akzeptanzquote der einzelnen
Review-Kommentare liegt bei **~30%**

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 learnable patterns exist between code reviews and their corresponding feedback. Our results highlight the current state of CodeRabbit's agentic code reviews, showing opportunity gaps for improvement, as well as shortcomings hindering its effectiveness.

CCS Concepts: • **Software and its engineering** → **Software creation and management**; **Software maintenance tools**; • **Computing methodologies** → **Artificial intelligence**

Additio

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

KI-Reviews funktionieren nicht besonders gut

- Akzeptanzrate meist $\sim \frac{1}{3}$
- Ohne “Human in the Loop” lassen sich hilfreiche und schädliche Kommentare nicht unterscheiden

KI-Reviews funktionieren nicht besonders gut, sind aber trotzdem nützlich

- Akzeptanzrate meist $\sim \frac{1}{3}$
- Ohne "Human in the Loop" lassen sich hilfreiche und schädliche Kommentare nicht unterscheiden
- Mit "Human in the Loop" lassen sich hilfreiche und schädliche Kommentare unterscheiden
- KI-Reviews finden relevante Qualitätsprobleme, die ein Mensch leicht übersieht
- Menschliche Code-Reviews finden viele weitere Qualitätsprobleme nachdem KI-Code-Review gelaufen ist

Gehen KI-Reviews nicht besser?
→ **“Agentische 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

- ☒ Code search ?
- ☒ Dependency lookup ?
- ☐ Commit history ?
- ☐ Finding descriptions ?
- ☐ Existing findings ?
- ☒ Added findings ?
- ☐ Work items ?

Attachments

Merge Request Info

Name

Merge Request Info

Ticket Info

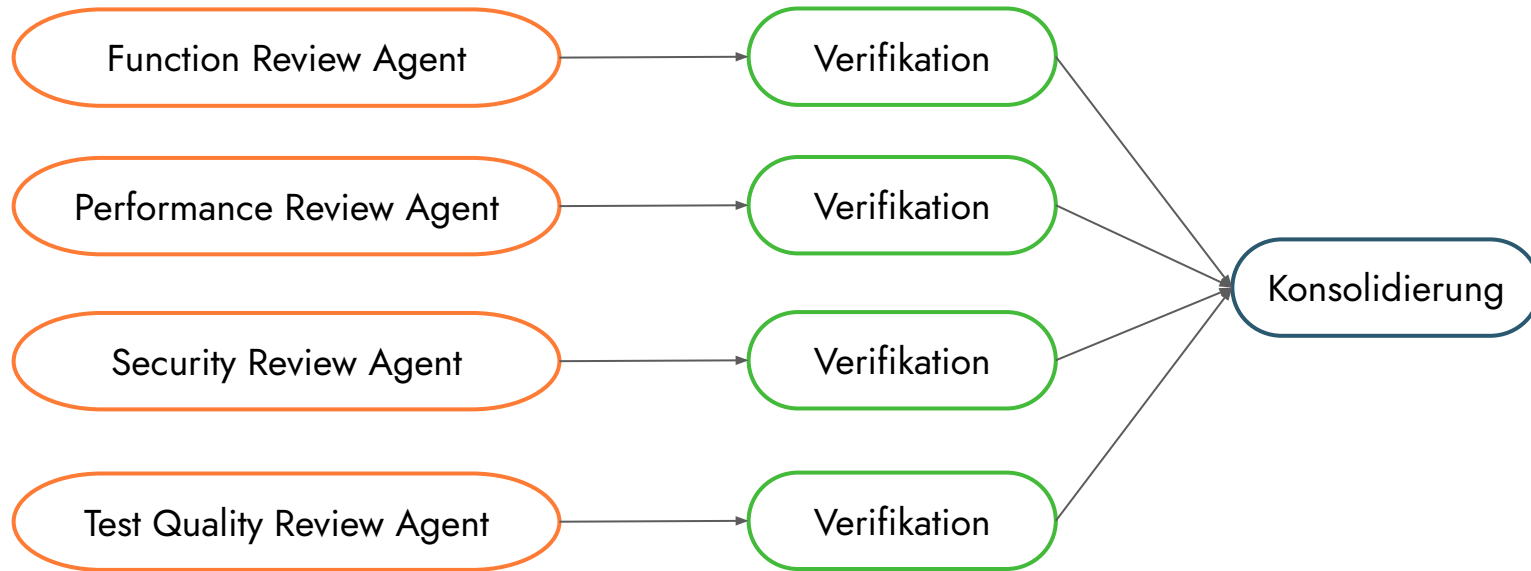
Name

Ticket Info

☐ Include ticket metadata ?

+ Add attachment

Pipeline agentischer Reviewer



154 Absences

Actions

Request to merge [absences](#) into [master](#)
This change contains 5 commits and introduces 27 new findings

STATUS

OPEN

VOTE

SKIPPED

BUILD JOBS

COMPLETE

AUTHOR

 Fabian Streitl

Commits **5** Findings **27** **1** Test Gaps **9** Test Gaps (75%) Test Coverage Coverage of Changes **42.5%** Impacted Tests **AI Review**

Functional Review **+4 more** **Generate new Review**

Review Result

This MR introduces a new Absence feature (entity, controller, DB access methods, UI controller, and seed fixtures), largely mirroring the existing Vacation feature. Functional review found the same input-validation gaps as Vacation (unvalidated year/month causing 500s, missing title length validation) plus new issues specific to the team overview page: a correctness bug where future-dated team assignments incorrectly affect "current" membership, and hidden employees leaking into the UI. Performance review flagged that both the team-membership query and the absence/vacation fetches for the overview page load far more data than needed (whole-table scans reduced in memory) under a single-connection DB lock, which could become a real bottleneck as the assignment table grows. Security review identified a CSRF-style risk on the new write endpoints (no origin/token protection) and unhandled date-parsing exceptions bypassing the intended validation error path. Test-quality review found no tests were added despite new test-only infrastructure, ambiguous/misleading seed-date construction, time-dependent fixture design, and duplicated fixture setup code, plus a DTO lacking accessors needed for meaningful test assertions. Several minor code-hygiene items (duplicated permission check methods, duplicated date-conversion logic, stale javadoc) round out the findings. Overall, the feature is functionally reasonable but needs tightened input validation, corrected query semantics, hardening against forgery/malformed input, and—most notably—actual test coverage before merge.

Review generated using Review Agents Functional Review, General Code Review, Performance Review, Security Review, Test Quality Review - Cost \$13.0611 - Triggered by cloud-admin-hummel@cqse.eu on Aug 04 2026 21:04

Expand All Collapse All

[pdash-server/src/main/java/eu/cqse/pdash/controllers/entities/AbsenceController.java:51-61](#)

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 unmappped 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](#)

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](#)

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'

[pdash-server/src/main/java/eu/cqse/pdash/controllers/entities/IAbsenceController.java:17-32](#)

Show code

Review-
Zusammenfassung

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

75+ % Akzeptanzrate

Hohe Akzeptanz der KI-Review-Ergebnisse **im Praxis-Einsatz**

Modell-Vergleich

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

Kosten

~8 USD im Schnitt pro Review, **bis zu 47 USD**
~100 USD pro Tag und Team

Wichtigkeit menschlicher Reviews

Menschliche Reviews finden viele weitere Qualitätsprobleme im Anschluss an das agentische Code-Review.

Teuer,
Weniger gut skalierbar,
Langsam

z.B. je Pull Request

Menschliche Code-Reviews

Manuell, (eher) deterministisch,
finden semantische Probleme

z.B. je Pull Request

Agentische Code-Reviews

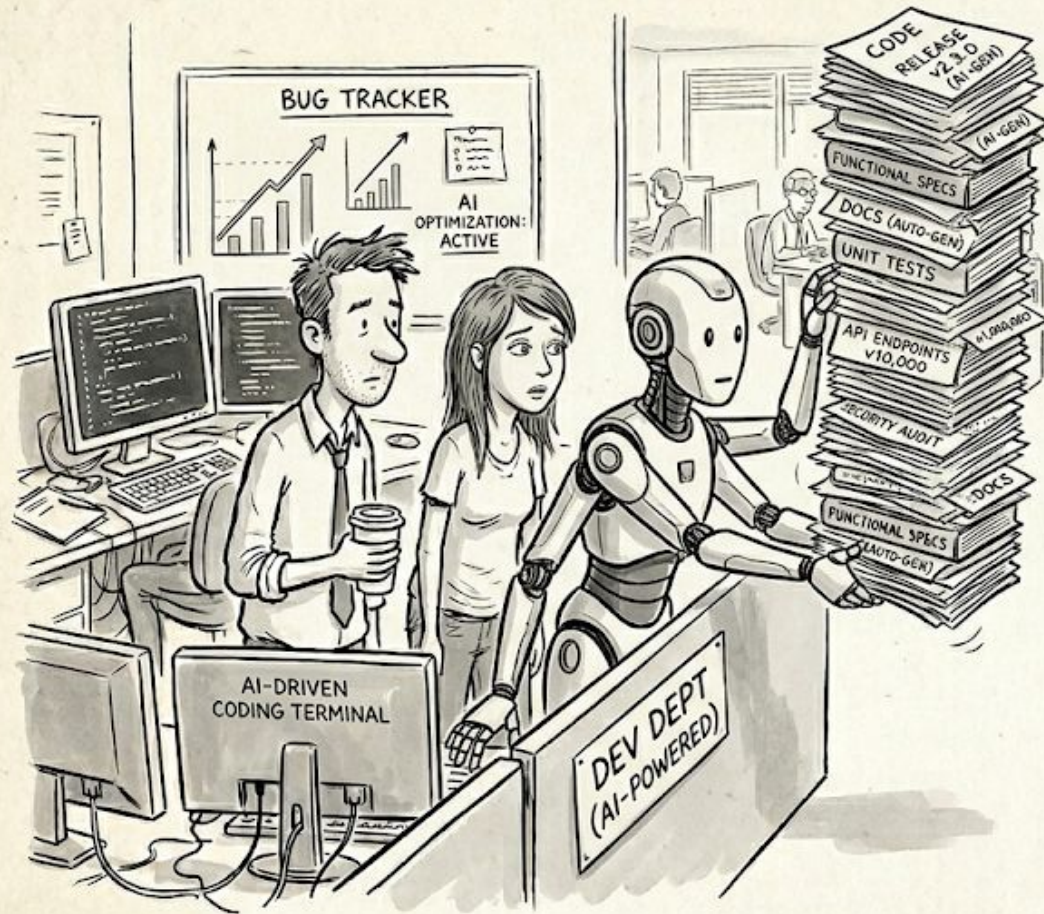
Automatisch, nicht deterministisch

z.B. je Commit

Automatisierte Analysen (Statische Analyse, Tests & Test-Gap-Analyse, ...)

Automatisch, deterministisch,
Finden Fehler- und Sicherheitslückenmuster, Wartbarkeitsprobleme, Vorgabenverletzungen
und (Test-) Fehler

Günstig,
Gut skalierbar,
Schnell



Wie skalieren Code Reviews?



Wie bringt man Coding-Agenten dazu, hohe Qualität zu liefern?

Wie sieht ein QS-Prozess für agentische Entwicklung aus?

Heise

“Aufschreiben” von Qualitätsvorgaben

Qualitätsvorgaben in CLAUDE.md

Non-Standard Code Conventions

- Java/Kotlin: Prefix enums with `E` and interfaces with `I`
 - Root packages: `com.teamscale.*`, `org.conqat.*`, `eu.cqse.*`
- C#: Prefix enums with `E` and interfaces with `I`, namespaces use `Cqse.{Product}.{Module}`
- TypeScript/React: Hooks use `Use` prefix (e.g., `UseLocalStorage.tsx`)

Company Standards

- Versioning: Year-based `v{YYYY.major.minor}`
- Null annotations: JSpecify Framework for null handling

Important ADRs

Review these architecture decisions in `architecture-decisions/`:

- 0007: TGA issues and branches
- 0013: Commit layering and branching store
- 0014: Storage architecture
- 0019: Project IDs
- 0025: Service layer versioning

Automatisierte, deterministische Qualitätsleitplanken als Teil des Agenten-Harness

Beispiele für Qualitätsleitplanken

- Ist der KI-Code „sauber“ und frei von typischen Fehlern und Sicherheitslücken?
→ Statische Analyse
- Benötigen wir neue/mehr Tests?
→ Test-Gap-Analyse
- Laufen die Regressionstests noch?
→ Ggf Testselektion

Gute Nachrichten

- Das ist alles schon recht gut erforscht/verstanden
- Es gibt Werkzeuge in diesem Umfeld

Das muss “nur noch” in den Agenten-Harness integriert werden

Beispiel 1:

Statische Analyse 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.

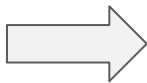
● Write(pdash-server/src/main/java/eu/cqse/pdash/controllers/PhotosController.java)
  | Added 74 lines, removed 12 lines
    20 import java.io.IOException;
    21 import java.net.URLEncoder;
    22 import java.nio.charset.StandardCharsets;
    23 +import java.nio.file.Files;
    24 +import java.nio.file.Paths;
```

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(EMAIL_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

```
● Compilation succeeded. Now running the teamscale findings check.
● Bash(ts-agent-helper coverage for-file pdash-server/src/main/java/eu/cqse/pdash/controllers/PhotosController.java 2>&1)
  └─ 0.0%
     └─ Allowed by auto mode classifier
● Coverage is 0% on this file - there are no tests. I'll proceed with findings check but flag this; for new logic in a
  0%-coverage file, refactoring risk is real but I can't reasonably add tests without significant mocking infrastructure for the
  HTTP calls.
● Bash(ts-agent-helper findings list pdash-server/src/main/java/eu/cqse/pdash/controllers/PhotosController.java 2>&1)
  └─ []
     └─ Allowed by auto mode classifier
● Findings list shows no committed findings, but I need to check what my local changes introduced. Let me use pre-commit
  analysis.

  Called plugin:teamscale:teamscale (ctrl+o to expand)
● Method length violation. Let me refactor to extract the photo fetch.
```

Behebung des Findings durch Extraktion einer Methode

```
        LOGGER.log(Level.WARNING, "Could not obtain user token", e);
        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
100
101
102
>> 103
104
105
106
107
108
109
110
111
112
>> 113
114
115
116
117
118
```

```
        return fetchPhoto(normalizedEmail, token);
    }

    private Response fetchPhoto(String email, String token) {
        HttpGet request = new HttpGet(ENDPOINT + email);
        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(email);
        }
    }
}
```

Erneute Prüfung

- Compilation clean, Teamscale pre-commit reports 0 findings.

Summary of changes to `pdash-server/src/main/java/eu/cqse/pdash/controllers/PhotosController.java:62`:

- Normalizes email to lowercase, returns 403 for non-@cqse.eu addresses.
- `DEV_OVERRIDE_PHOTO_PATH` env var short-circuits to a local file (content type via `Files.probeContentType`).
- `DEV_FIXED_AUTH_TOKEN` env var skips the GCP metadata token call.
- On Dec 24, tries `/home/santa/photo.svg` and falls through to the normal flow if the file is missing (so locally nothing changes).
- Extracted `fetchMetadataToken`, `fetchPhoto`, `readLocalPhoto`, `tryReadLocalPhoto`, `isChristmasEve` to keep `get(...)` under the 30-SLOC method-length threshold.

Teamscale findings: fixed 1 (method length), tolerated 0, skipped 0. Coverage on this file is 0% – these changes are unverified by tests; manual exercise of each branch is recommended before deploy.

Agenten mit Statische Analyse-Leitplanke sind besser*

Leitplanken auf Basis statischer Analyse verhelfen LLMs zu einer erheblich verbesserten Codequalität, die über funktionale Korrektheit hinausgeht.



Sicherheitsprobleme

Sicherheitsrelevante Schwachstellen im Code

Vorher vs. Nachher

>40 % → 13 %



Lesbarkeitsprobleme

Mängel in Struktur und Verständlichkeit

Vorher vs. Nachher

>80 % → 11 %



Zuverlässigkeitsprobleme

Fehleranfälligkeit und Robustheitsmängel

Vorher vs. Nachher

>50 % → 11 %

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

Beispiel 2:

Test-Gap-Analyse in Claude Code

Test Gaps für aktuellen Pull Request

158 Progressive loading

Actions

Request to merge [progressive-loading](#) into [master](#) OPEN COMMENTED

AUTHOR BH Benjamin Hummel

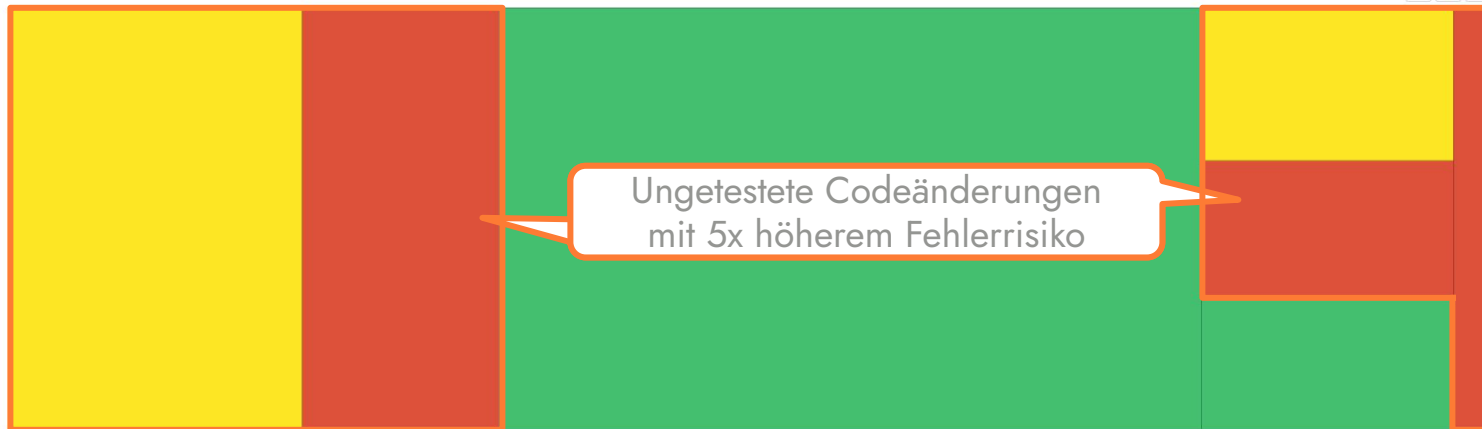
This change contains 1 commit and introduces 4 new findings

▼ Relevant Build Jobs 1

^ Test Gaps 5 Test Gaps (71%)

May 05 2026 14:30 - Now | Test Gap: 71.4%

Coverage sources All



Aufruf

```
Claude Code v2.1.138
Opus 4.7 (1M context) with xhigh effort · Claude Team
~/git/pdash

> /teamscale:pr-close-test-gaps

● I'll start by fetching test-gap information for the current branch.

● Bash(ts-agent-helper test-gaps for-pr)
  | ts-agent-helper: resolved as pull request cqse/internal/pdash/158
  |   source=progressive-loading:HEAD target=master:HEAD
  |   Uniform Path;Method Name;Method Region Lines;Test State
  |   ... +10 lines (ctrl+o to expand)
  |   | Allowed by auto mode classifier

● Resolved to PR cqse/internal/pdash/158 (progressive-loading vs master). Let me look at the source files
  and existing tests to understand the context.

  Read 3 files (ctrl+o to expand)

● Let me explore existing tests to understand the framework and conventions.

  Searched for 2 patterns (ctrl+o to expand)

● Bash(find /home/hummel/git/pdash/pdash-server/src/test -name "*Test.java" | xargs ls -la 2>/dev/null |
  wc -l && find /home/hummel/git/pdash/pdash-server/src/test -pa...)
  | 17
  | | Allowed by auto mode classifier

  Read 1 file (ctrl+o to expand)

● Let me look at more existing tests to understand the controller and aggregated model testing patterns.

  Searched for 9 patterns, read 9 files, listed 1 directory (ctrl+o to expand)

  Searched for 7 patterns, read 2 files (ctrl+o to expand)

● Now I'll write the RichProject test.
```

Behebung von Test Gaps

```
... +25 lines (ctrl+o to expand)
└─ (timeout 5m)
└─ Allowed by auto mode classifier
```

● All 4 new tests pass.

● Summary

Resolved as [pull request cqse/internal/pdash/158](#) (progressive-loading vs master).

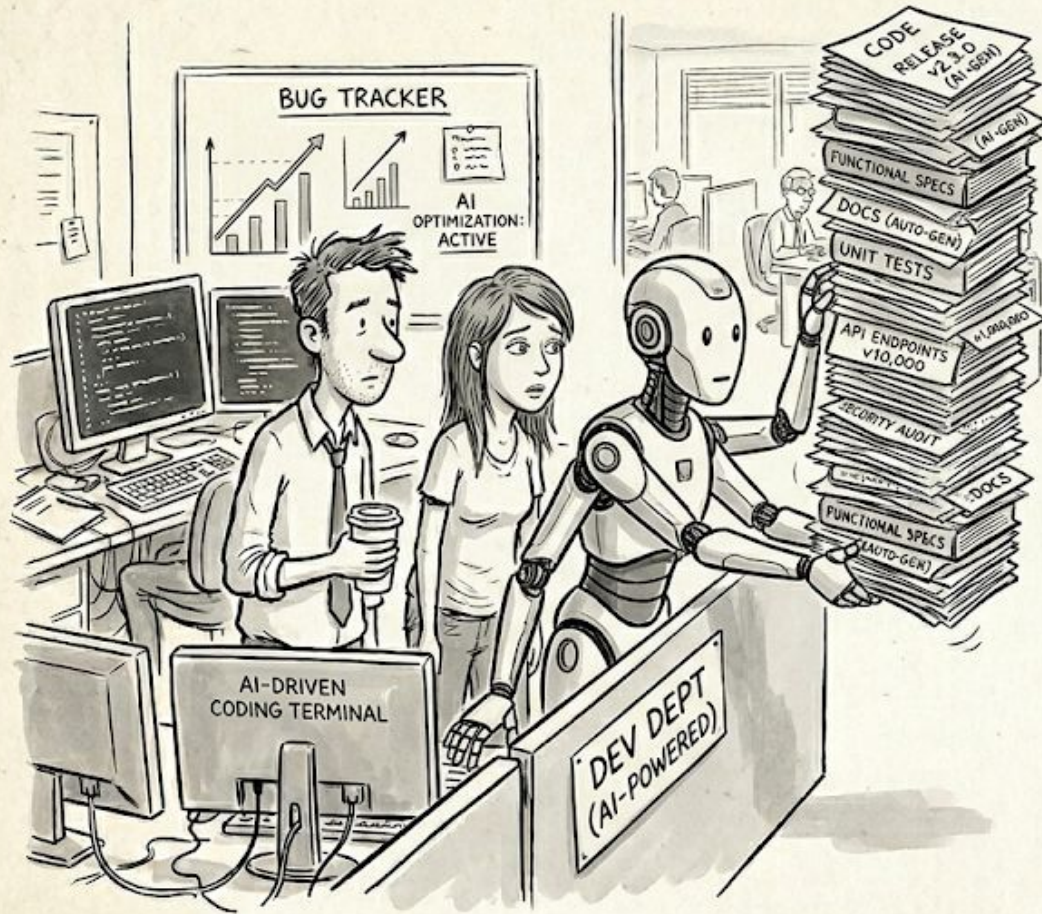
Gaps closed (4 tests, all passing)

- `pdash-server/src/test/java/eu/cqse/pdash/models/aggregated/RichProjectTest.java` – covers `RichProject.forProjectLogView` (untested addition) and indirectly the new `RichTask(Task, int, Team)` constructor (untested addition):
 - `forProjectLogViewAggregatesLoggedMinutesAndSkipsTimeLogs` – verifies the factory aggregates per-task and per-tasklist logged minutes, and that individual `TimeLogs` are *not* loaded (lazy-loading contract).
- `pdash-server/src/test/java/eu/cqse/pdash/controllers/ui/projects/ProjectLogUiControllerTest.java` – covers `ProjectLogUiController.taskTimeLogs` (untested addition):
 - `taskTimeLogsRendersTimeLogsForMatchingTask` – 200 OK with HTML containing the time log notes and the formatted minutes (135 min → 2:15).
 - `taskTimeLogsReturns404WhenTaskBelongsToDifferentProject` – 404 when the task ID exists but belongs to a different project.
 - `taskTimeLogsReturns404WhenTaskDoesNotExist` – 404 when the task ID does not exist at all.

Gaps skipped

- `RichProject 7-arg constructor` (`RichProject.java:83-88`) – flagged as "untested modification", but the change is purely a 2-line delegation to the new private constructor with `loadTimeLogs=true`. Its behavior is already exercised by existing `HubSpotControllerDealUpdateTest` usages of new `RichProject(syncAndGetDeal())` (which chains through it).
- `ProjectLogUiController.buildPageModel` (`ProjectLogUiController.java:136-161`) – private helper called from `page()` and `tasks()`; its change was swapping eager loading for `forProjectLogView`. The aggregation behavior it relies on is now directly covered by `RichProjectTest`; the rest of the method is just `Map` population without behavior worth a separate HTTP-level test.

*Brewed for 4m 26s



Wie skalieren Code Reviews?



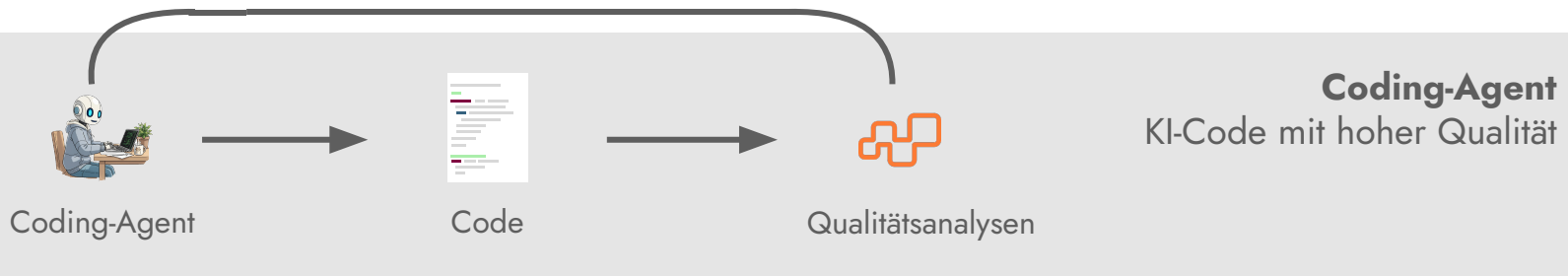
Wie bringt man Coding-Agenten dazu, hohe Qualität zu liefern?

Wie sieht ein QS-Prozess für agentische Entwicklung aus?

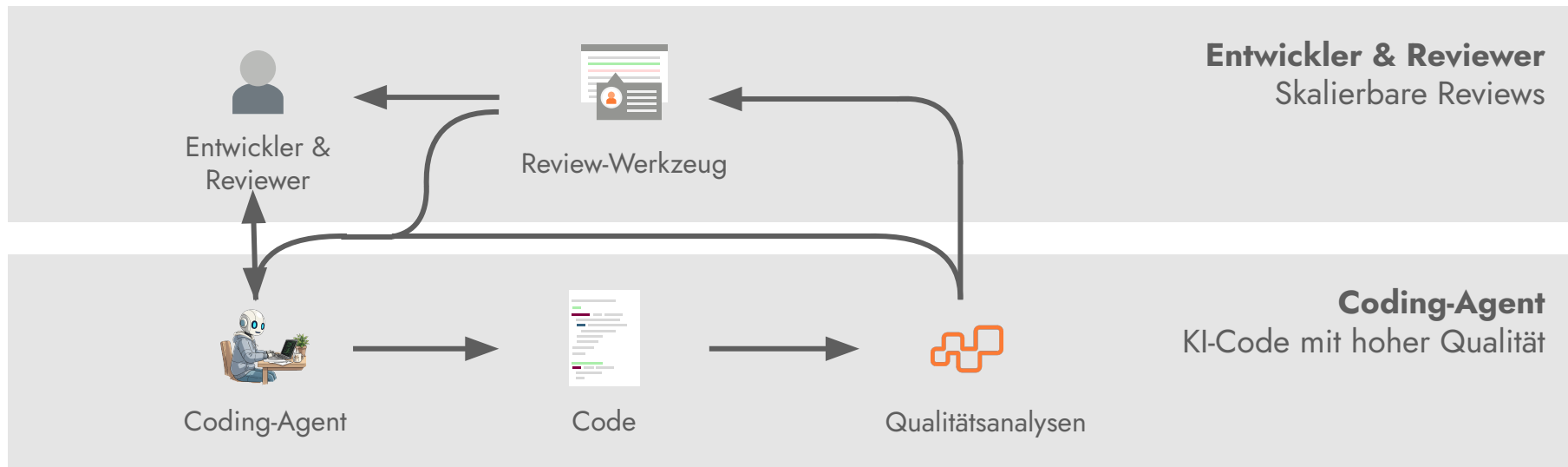
Heise

Automatisierte QS-Maßnahmen

Statische Analyse, Testautomatisierung, KI-Reviews, ...



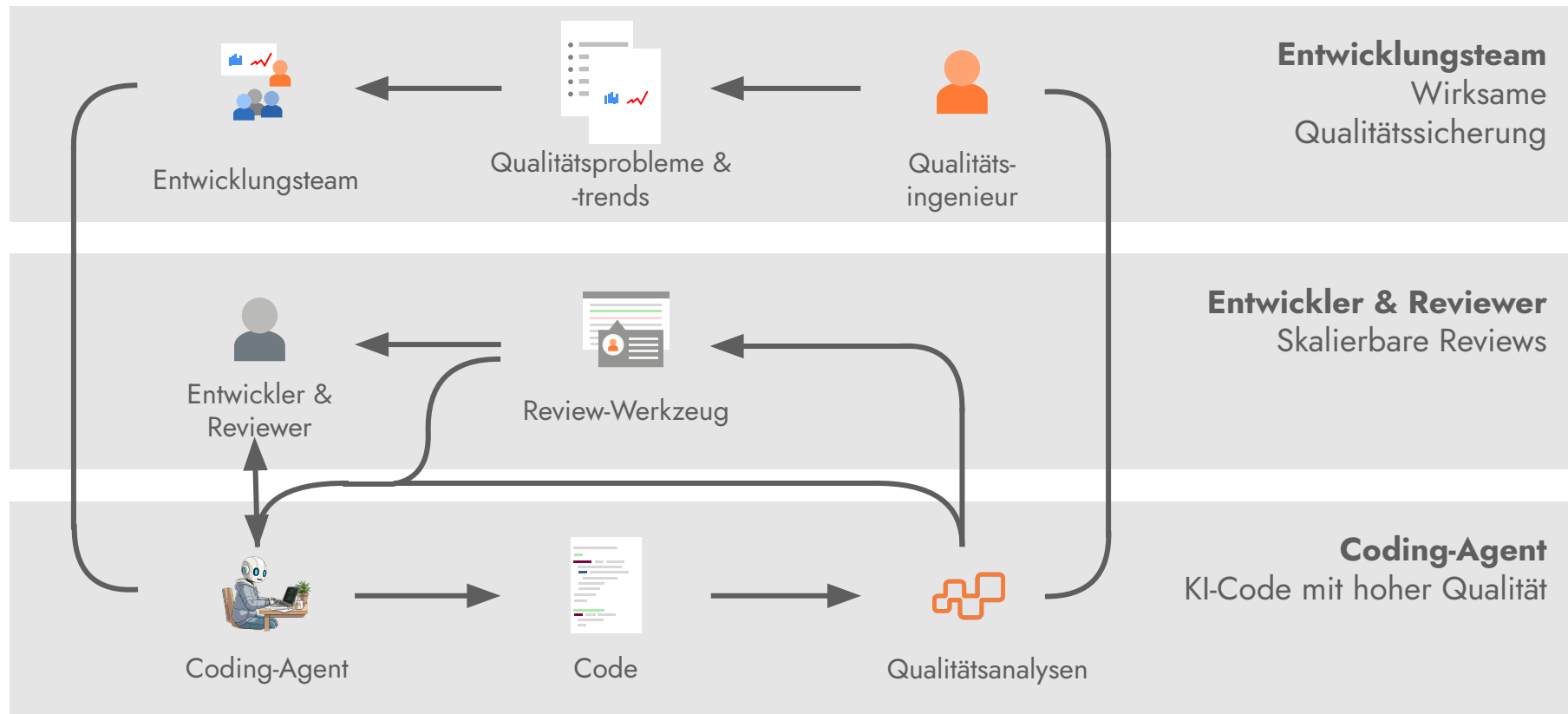
Automatisierte QS-Maßnahmen
Statische Analyse, Testautomatisierung, KI-Reviews, ...



Automatisierte QS-Maßnahmen

Statische Analyse, Testautomatisierung, KI-Reviews, ...

QS-Prozess für agentische Entwicklung



Automatisierte QS-Maßnahmen

Statische Analyse, Testautomatisierung, KI-Reviews, ...

Retrospektive zu Coding-Agenten & Qualität



Reflektion

Identifikation von Best und Bad Practices



Vorgabendefinition

Gemeinsame Definition von Qualitätsvorgaben



Wartung des QS-Setups

Konfiguration und Weiterentwicklung von agentischen Reviewern und Agenten-Harness



Drift-Monitoring

Analyse von Qualitätstrends des Gesamtsystems, ggf. Ableitung von Gegenmaßnahmen



Wissensaufbau

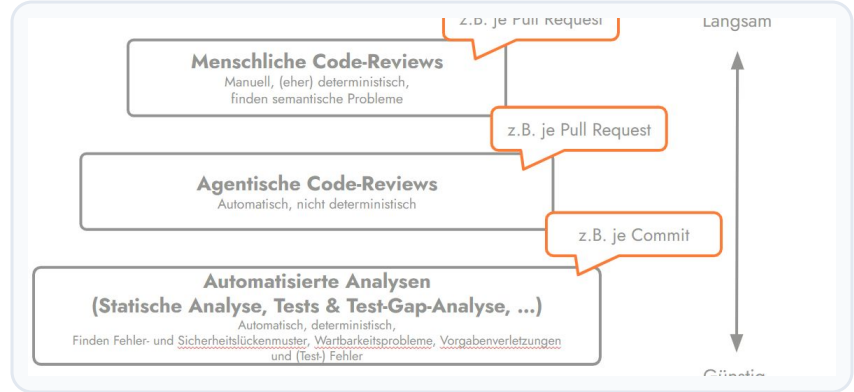
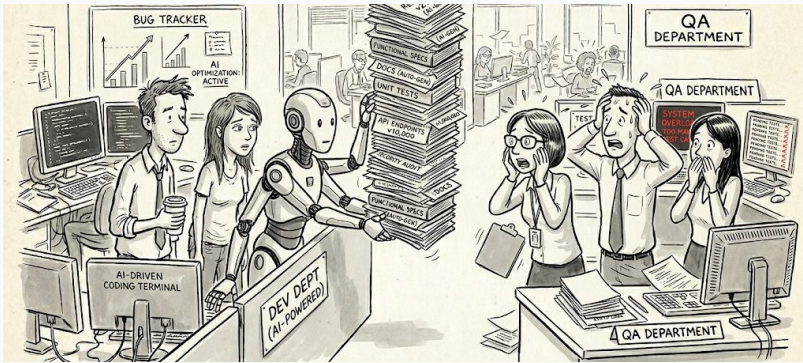
Kontinuierlicher Austausch zu Agenten-Einsatz, -konfiguration und -QS



Kollaborative, strukturierte Qualitätssicherung beim Einsatz von Coding-Agenten

Die Retrospektive fördert das gemeinsame Agenten- und Qualitätsverständnis und stellt sicher, dass Maßnahmen direkt im QS-Setup bzw. Entwicklungsalltag verankert werden.

Zusammenfassung



```

● Bash(ts-agent-helper test-gaps for-pr)
  └─ ts-agent-helper: resolved as pull request cqse/internal/pdash/158
     source=progressive-loading:HEAD target=master:HEAD
     Uniform Path;Method Name;Method Region Lines;Test State
     ... +10 lines (ctrl+o to expand)
     └─ Allowed by auto mode classifier

● Resolved to PR cqse/internal/pdash/158 (progressive-loading vs master). Let me look at the source files
  and existing tests to understand the context.

  Read 3 files (ctrl+o to expand)

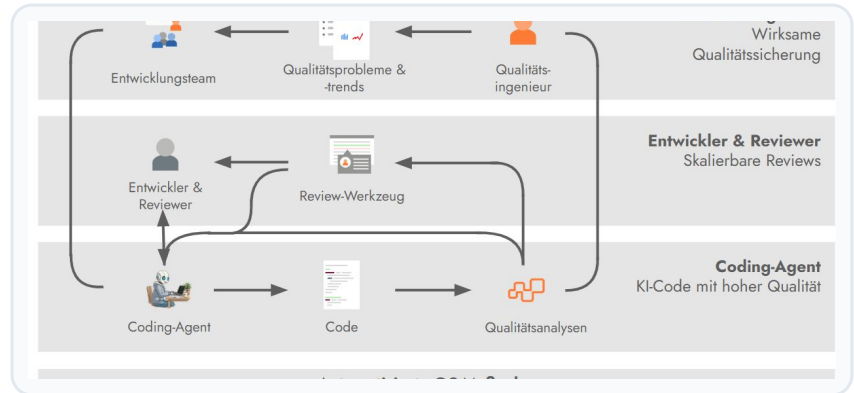
● Let me explore existing tests to understand the framework and conventions.

  Searched for 2 patterns (ctrl+o to expand)

● Bash(find /home/hummel/git/pdash/pdash-server/src/test -name "*Test.java" | xargs ls -la 2>/dev/null |
  wc -l && find /home/hummel/git/pdash/pdash-server/src/test -pa...)
  └─ 17
     └─ Allowed by auto mode classifier

  Read 1 file (ctrl+o to expand)

  Let me look at more existing tests to understand the controller and associated model testing patterns.
    
```



Danke fürs Zuhören

Hier gibt's die Folien und Updates



Wir bitten um dein Feedback

Jetzt bewerten!



entwickler.de

Studien & Quellen

- Developer Ecosystem Survey 2026 von JetBrains
<https://blog.jetbrains.com/research/2026/08/ai-coding-agent-adoption-2026/>
- The Acceleration Whiplash von Faros
<https://www.faros.ai/research/ai-acceleration-whiplash>
- DORA State of AI-assisted software development von Google
<https://dora.dev/dora-report-2025/>
- He et al.: Speed at the Cost of Quality, MSR 2026
<https://arxiv.org/pdf/2511.04427>
- Denisov-Blanch: Can you prove AI ROI in Software Engineering (at 4:00/ 13:30 mins)
<https://www.youtube.com/watch?v=JvosMkuNxF8>