LU10a - Ablauf eines Codereviews

Auf einen Blick
Ein formales Codereview (Inspektion) läuft in sechs Phasen ab: **Planung, Kick-off, Vorbereitung, Reviewsitzung, Nacharbeit, Follow-up**. Die Gutachter suchen Mängel und halten sie als **Befunde** fest. Verbessert wird erst danach: Der Autor behebt die Befunde in der Nacharbeit, der Moderator prüft im Follow-up, ob alles erledigt ist.

Lernziele

Sie können

- die Phasen eines Codereviews und die Rollen beschreiben,
- als Gutachter:in Befunde druckreif formulieren und bewerten,
- erklären, warum in der Reviewsitzung nicht diskutiert und nicht verteidigt wird.

Rollen

Rolle	Aufgabe
Moderator:in	plant das Review, leitet die Sitzung, achtet auf die Regeln, prüft im Follow-up die Korrekturen
Autor:in	hat den Code geschrieben, protokolliert in der Sitzung die Befunde, behebt sie in der Nacharbeit
Gutachter:in (2–4 Personen)	prüft den Code in der Vorbereitung und trägt die Befunde in der Sitzung vor

Als Gutachter eignen sich besonders erfahrene Entwickler:innen. Weniger erfahrene lernen im Review aber ebenso viel, weil sie fremden Code genau lesen.

Die sechs Phasen

1. Planung

Die Moderatorin legt fest, **welcher Code** geprüft wird, und fragt die Gutachter an. Der Umfang muss überschaubar sein: Wer zu viel Code auf einmal prüfen soll, liest nicht mehr gründlich.

2. Kick-off

Kurze Besprechung mit allen Beteiligten:

- Ziele und Termine festlegen,

- Code und Unterlagen verteilen (z.B. Codingstandards, Checkliste),
- bei umfangreichen Standards festlegen, wer welche Aspekte prüft. Jeder Aspekt wird von mindestens zwei Gutachtern geprüft.

3. Vorbereitung

Jede Gutachterin prüft den Code **allein** und notiert jeden Befund:

Dateiname	Zeile	Bewertung	Befund	Gutachter
ExamService.py	97	N	Die Variable `oldExam` ist in camelCase geschrieben. Gemäss Codingstandard 01 muss sie `old_exam` heissen.	AB

Bewertung:

- **H = Hauptfehler:** Muss sofort korrigiert werden (falsches Verhalten, Sicherheitsproblem).
- **N = Nebenfehler:** Sollte bei Gelegenheit korrigiert werden (Verstoss gegen Standard, schlechte Wartbarkeit).
- **B = Bemerkung:** Kann korrigiert werden, ist aber nicht zwingend.

Der Befund ist bereits **druckreif** formuliert. So kann er in der Sitzung direkt ins Protokoll übernommen werden.

4. Reviewsitzung

Die Moderatorin geht die Befunde sortiert nach Datei und Zeile durch. Die Gutachter tragen ihre Befunde vor, die Gruppe einigt sich auf Formulierung und Bewertung, der Autor schreibt das Protokoll.

Regeln:

- Die Gutachter tragen **nur Befunde** vor. Sie suchen in der Sitzung keine Lösungen und kritisieren den Code, nicht die Person.
- Der Autor **verteidigt sich nicht** und kommentiert die Befunde nicht. Er fragt nur nach, wenn eine Formulierung unklar ist.
- Bei unterschiedlichen Bewertungen entscheidet die Begründung, nicht die Mehrheit. Die Moderatorin fasst zusammen.
- Am Schluss entscheidet die Gruppe: **angenommen**, **angenommen mit Nacharbeit** oder **neues Review nötig**.

Nach der Sitzung kann ein informeller Teil folgen («die dritte Stunde»): Jetzt darf diskutiert werden, und der Autor holt sich Ratschläge, wie er die Befunde beheben kann.

5. Nacharbeit

Der Autor behebt die Befunde, zuerst die Hauptfehler. Zu jedem Befund hält er fest, was er geändert hat – oder begründet, warum ein Befund nicht umgesetzt wird.

6. Follow-up

Die Moderatorin prüft, ob alle Befunde erledigt sind. Bei vielen Hauptfehlern wird ein neues Review angesetzt.

Gute und schlechte Befunde

schlecht	besser
«Das ist unschön.»	«`save` (Zeilen 87–130) lädt Daten, erstellt das Objekt und speichert es. Die Methode hat mehrere Aufgaben (SRP).»
«Falscher Name.»	«Die Variable `oldExam` (Zeile 97) ist nicht in snake_case. Richtig: `old_exam`.»
«Wer hat das geschrieben?»	«`except Exception` (Zeile 31) fängt jeden Fehler ab, auch Programmfehler.»

Ein guter Befund nennt **Ort**, **was** nicht stimmt und, wo nötig, **gegen welche Regel** er verstösst.

Checkliste für Gutachter

Zusätzlich zu den Codingstandards:

- Tut der Code, was er soll? Gibt es Fälle, die nicht behandelt werden (leere Liste, None, Fehler)?
- Werden Fehler sinnvoll behandelt oder still verschluckt (`except: pass`)?
- Sind Namen verständlich? Hat jede Funktion genau eine Aufgabe? Gibt es doppelten Code?
- Gibt es Magic Numbers oder fest eingebaute Werte?
- Passen HTTP-Statuscodes, Rückgabewerte und Docstrings zum tatsächlichen Verhalten?
- Gibt es Sicherheitsprobleme (Passwörter im Code, fehlende Berechtigungsprüfung)?
- Gibt es Tests für den Code?

Kontrollfragen

1. Nennen Sie die sechs Phasen eines Codereviews.
2. Warum soll der Autor in der Sitzung seinen Code nicht verteidigen?
3. Wann ist ein Befund ein Hauptfehler, wann ein Nebenfehler?
4. Wer behebt die Befunde, und wer prüft, ob sie behoben sind?

M450-LU10, M450-D1I, M450-D1A

Marcel Suter

From:
https://wiki.bzz.ch/ - **BZZ - Modulwiki**

Permanent link:
https://wiki.bzz.ch/modul/m450/learningunits/lu10/codereview

Last update: **2026/11/23 09:21**