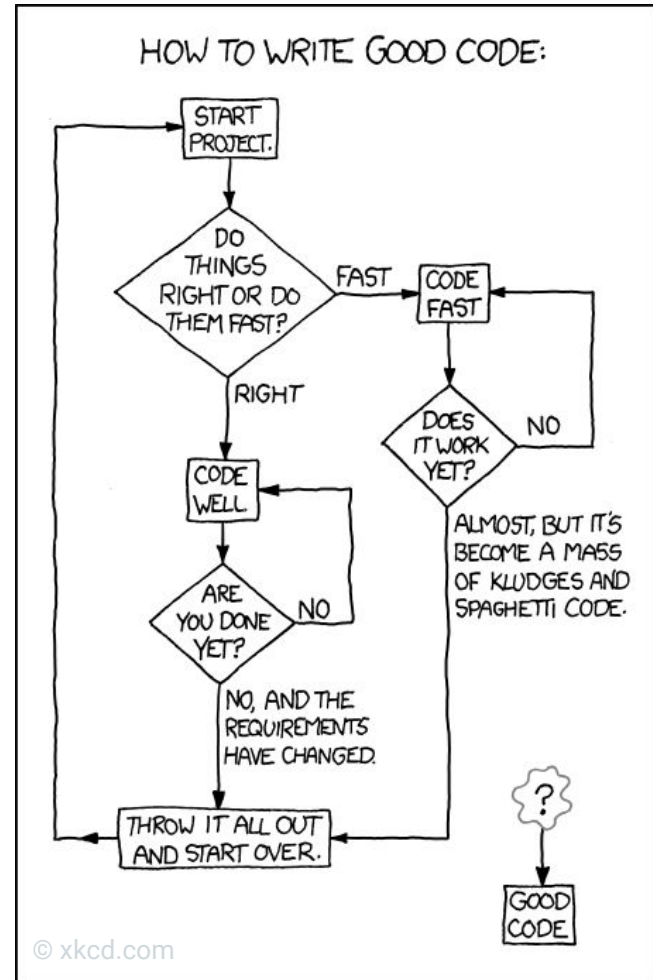


How to Code Review

Kannste schon so machen,
aber dann isses halt...

twitter.com/ja_sie | jasie.de/code-review



Übersicht

- I. Einführung
- II. Review Aspect: **Tests**
- III. Review Aspect: **Performance**
- IV. Review Aspect: **Data Structures**
- V. Review Aspect: **Coding Principles**
- VI. Review Aspect: **Security**
- VII. Code of Conduct

I) Einführung

Begriff / Konzept

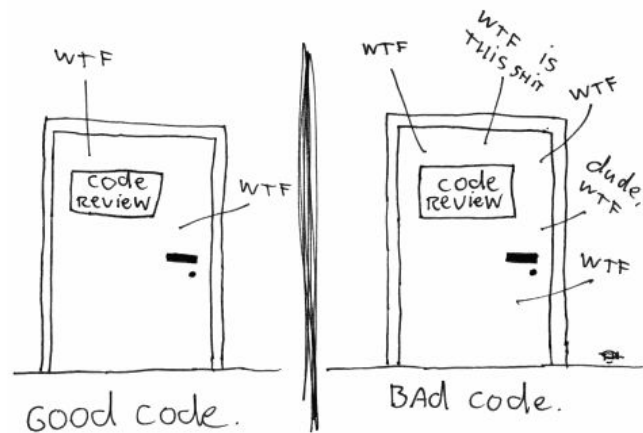
Code Review aka Code Inspection

=

systematische, manuelle Untersuchung von Quellcode durch eine oder mehrere Personen

Ziel:
Verbesserung der Softwarequalität

The ONLY valid measurement
OF code QUALITY: WTFs/minute



Historie

erster formeller Code Review Prozess:

Michael Fagan, IBM Systems Journal, 1976
*"Design and code inspections to reduce errors
in program development"*

"Fagan Inspections" / "Fagan Defect-Free Process"

Reviews and inspections

Michael Fagan
President, Michael Fagan Associates
Palo Alto, California, USA
Michael@mfagan.com



IBM T.J. Watson Research Lab:
senior technical staff member

IBM Quality Institute: Co-founder

Corporate Achievement Award
from IBM

University of Maryland:
Visiting Professor

Major contribution:
software inspection process

Current interests: improving
the Fagan Defect-Free Process

Contra & Pro

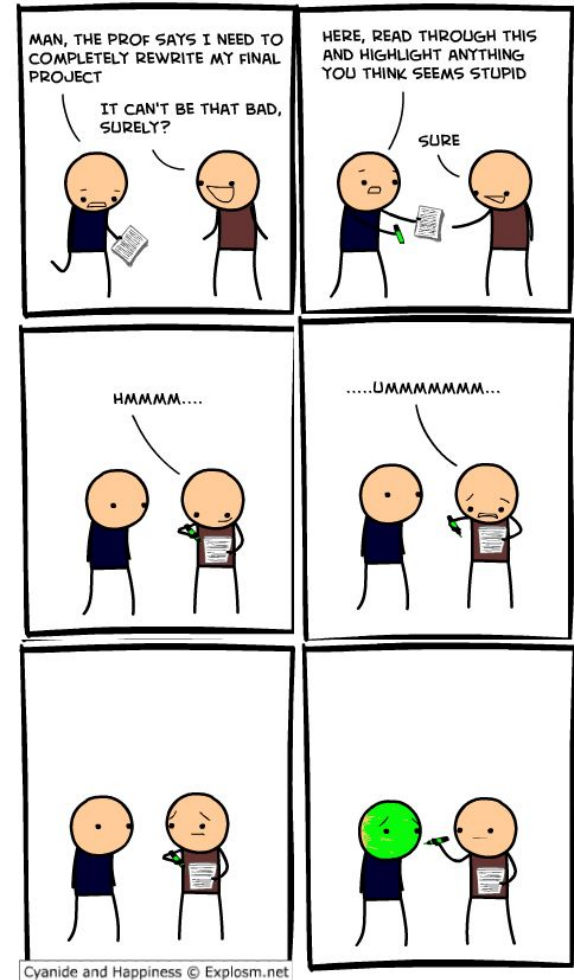
- zusätzlicher Aufwand (zeitlich & kognitiv)
- größeres Konfliktpotenzial
- + Verbesserung der Codequalität
- + Verteilung von Wissen
- + frühzeitige Fehlererkennung (und -beseitigung)



Ablauf (toolgestützt)

- 1) **Author** publiziert Codeänderungen
(z.B. Pull Request / Merge Request)
- 2) **Reviewer** begutachtet Codeänderungen
 - a) Approve ("LGTM") oder
 - b) Decline durch **Reviewer**
- 4) a) Merge in Codebase oder
b) Korrektur durch **Author**

ggf. rundenbasiert

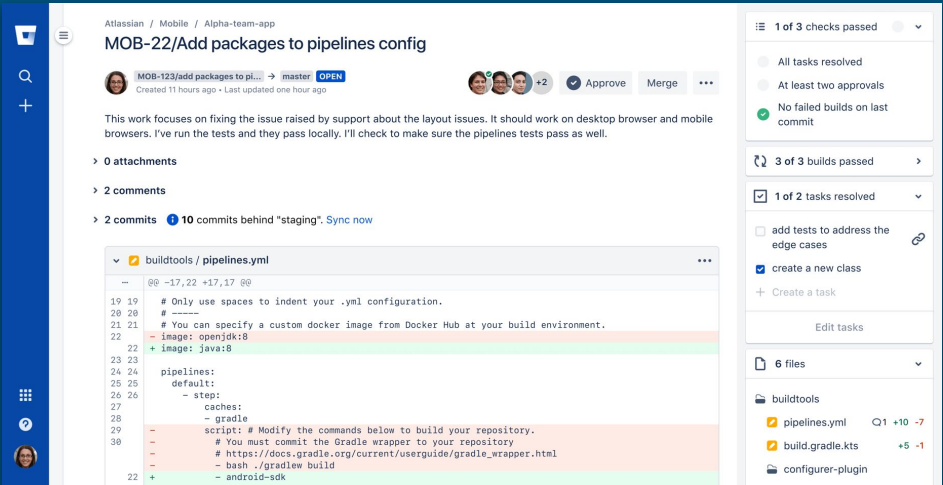
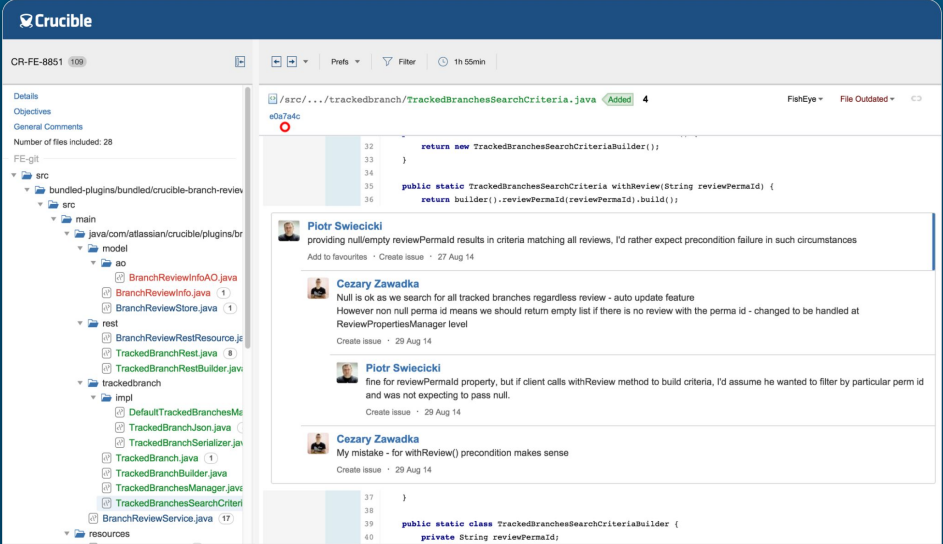


Review Tools

Vorteile:

- Vorher Nachher
- zeitliche Flexibilität
- Build Pipeline/CI

- GitHub Pull Requests
github.com/features/code-review
- Crucible
atlassian.com/software/crucible
- Bitbucket Code Review
bitbucket.org/product/de/features/code-review
- StackExchange Code Review
codereview.stackexchange.com/



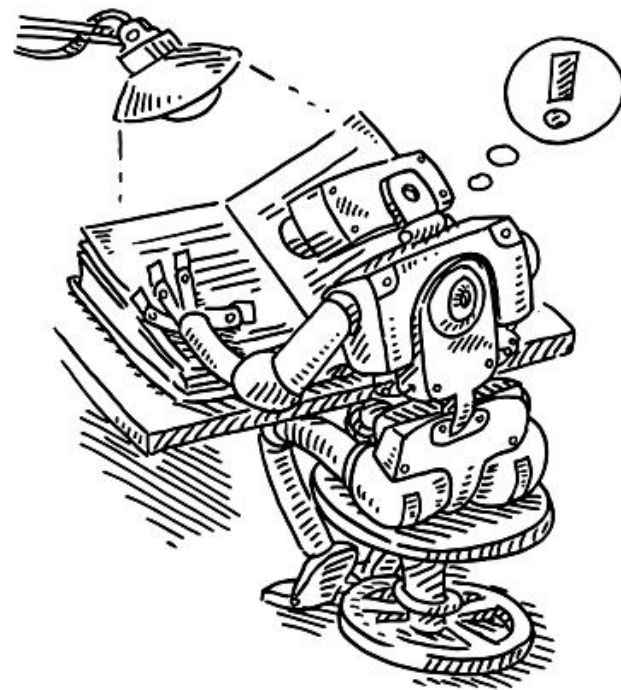
Gegenstand der Prüfung

- Formatting?
- Code Style?
- Naming Conventions?
- Test Coverage?

NEIN! weil automatisierbar

Was dann??

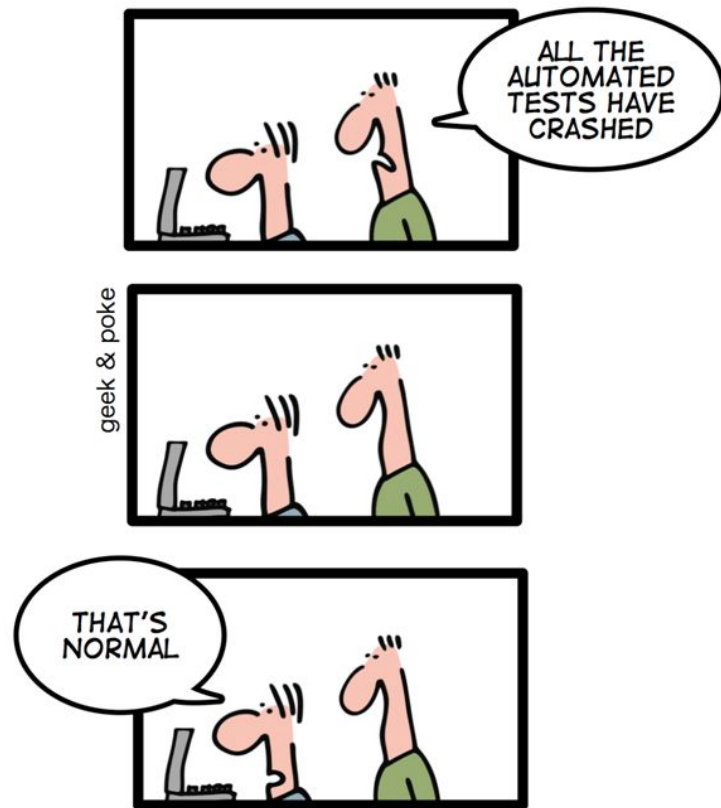
Was nicht automatisierbar ist!



II) Review Aspect: **Tests**

Tests /1

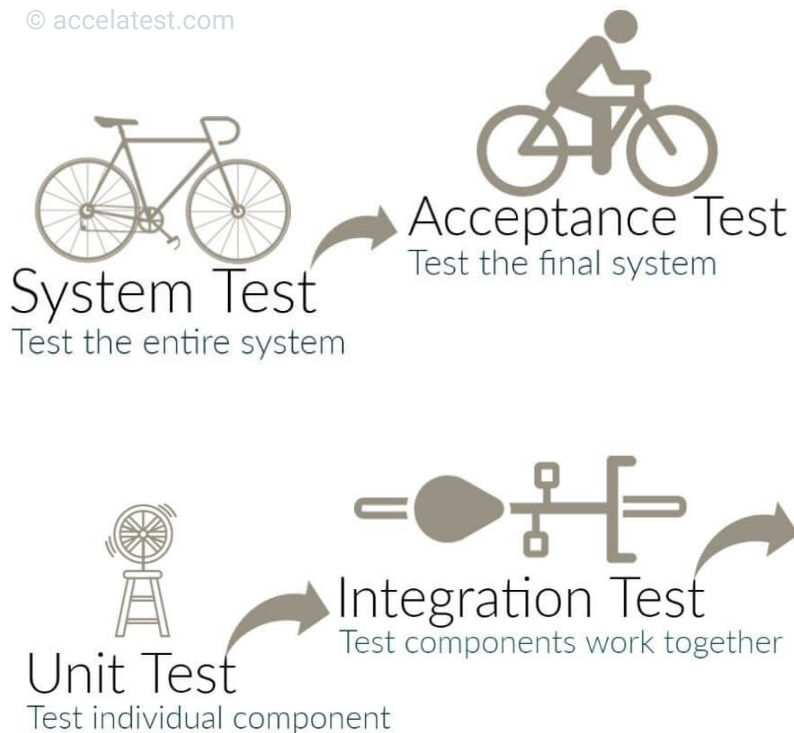
- ❑ Code durch autom. Test abgedeckt?
z.B. neue Interface function
- ❑ Tests human-readable (verstehbar)?
z.B. Naming, Structure, Assertion Begründung, Code Style
- ❑ Anforderungen abgedeckt?
z.B. Acceptance Criteria
- ❑ weitere Testfälle sinnvoll?
fehlende Anforderungen wie z.B. Validation



Tests /2

- ❑ Einschränkungen/Grenzfälle getestet?
z.B. 'Negative Tests', Limits, Filter
- ❑ richtiger Typ/ richtiges Level?
z.B. Unit vs. Integration Test
- ❑ Security berücksichtigt?
z.B. Endpoint protection
- ❑ Performance getestet?
z.B. Benchmarking, Lasttests

© accelatest.com



III) Review Aspect: **Performance**

Performance /1

- ❑ Performance-Anforderungen berücksichtigt und getestet?
z.B. instant-Aktualisierung des Filterergebnis'
- ❑ So wenig wie möglich, so viel wie nötig?
z.B. liefert Endpunkt nur genau das Benötigte aus
- ❑ Effizienter Code?
z.B. Locks auf geteilte Ressourcen, Reflections, Timeouts
- ❑ Memory-Risiken?
z.B. Leaks, Infinite Data



Performance /2

- ❑ Connections/Streams geschlossen?
z.B. Datenbankverbindung nur geöffnet
- ❑ Race Condition möglich?
z.B. zwei Nutzer, die dasselbe Item bearbeiten
- ❑ Caching Pitfalls?
z.B. falsche Cache Invalidation
- ❑ Datenstruktur performant?
z.B. beim Durchsuchen von Datensätzen



© Python (Monty) Pictures Limited

IV) Review Aspect: **Data Structures**

Data Structures /1

- ❑ **List zweckmäßig?**
AP z.B. wiederholte Suche eines bestimmten Items by ID oder Umsortierung der Items
- ❑ **Map zweckmäßig?**
AP z.B. Map als globale Konstante oder Umsortierung der Items oder Iteration über Items
- ❑ **Set zweckmäßig?**
AP z.B. Zugriff auf Items by position oder Duplicates-Verlust
- ❑ **Array zweckmäßig?**
AP z.B. hierarchische Daten abgebildet



Data Structures /2

- ❑ Stack zweckmäßig?

AP z.B.: nicht genutzt für push/pop Operationen (LIFO)

- ❑ Queue zweckmäßig?

AP z.B.: nicht genutzt für add to/remove (FIFO)

- ❑ Tree zweckmäßig?

AP z.B. jeder Knoten hat nicht mehr als ein Kind

- ❑ Heap zweckmäßig?

AP z.B. Suche nach anderen Items als dem größten/kleinsten

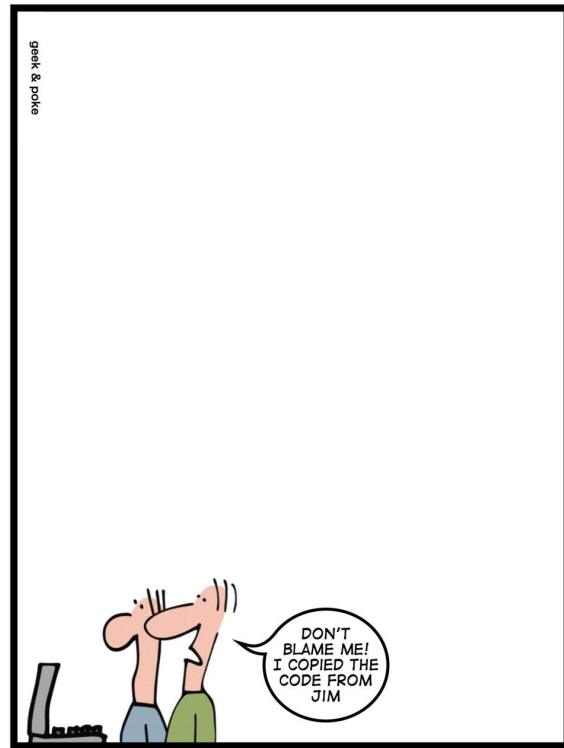


V) Review Aspect: **Coding Principles**

Coding Principles /1

- ❑ KISS (Keep It Simple, Stupid) verletzt?
z.B. Methode ist lang, komplex, schwierig zu verstehen
- ❑ DRY (Don't Repeat Yourself) verletzt?
z.B. duplicate Code
- ❑ SRP (Single Responsibility Principle) verletzt?
z.B. Methoden einer Klasse, die sich gleichzeitig ändern
- ❑ OCP (Open-Closed Principle) verletzt?
z.B. Serien von if-Statements, die einen bestimmten Typ durchgehen

RECENTLY DURING CODE REVIEW



SINGLE SOURCE PRINCIPLE

Coding Principles /2

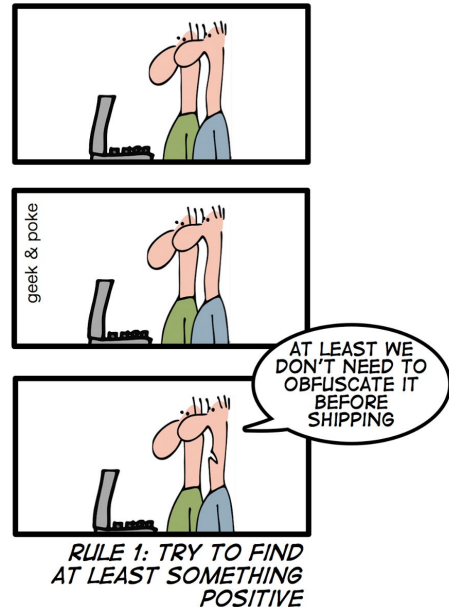
- ❑ LSP (Liskov Substitution Principle) verletzt?
z.B. explizites Casting
- ❑ ISP (Interface Segregation Principle) verletzt?
z.B. Interfaces mit vielen Methoden
- ❑ DIP (Dependency Inversion Principle) verletzt?
z.B. 'new' Keyword statt Dependency Injection
- ❑ YAGNI (You Aren't Gonna Need It) verletzt?
z.B. Funktion eingebaut, die nicht aufgerufen wird



Code Smells /1

- ❑ Lost Intent?
z.B. Name beschreibt nicht Zweck der Methode
- ❑ Long Param List?
z.B. zu viele Parameter übergeben
- ❑ Dead Code?
z.B. auskommentierter Code ohne TODO-Comment
- ❑ Lazy Method/Object / Middle Man
z.B. Methode tut zu wenig

HOW TO MAKE A GOOD CODE REVIEW



Code Smells /2

- ❑ Black Sheep?
z.B. Methode ganz anders als die anderen Class Methods
- ❑ Cyclomatic Complexity?
z.B. zu tiefe if/else Verschachtelung
- ❑ Irreführende Code Comments?
z.B. Methoden-Docu nicht aktualisiert
- ❑ Type “Unsafety”?
z.B. zu viele Return Type erlaubt

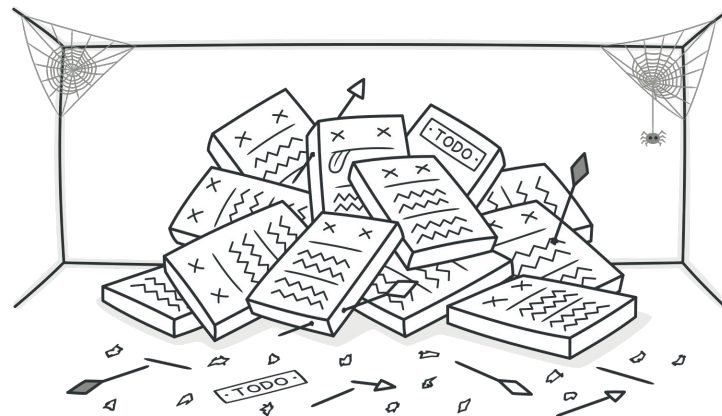


Quelle: reddit.com

VI) Review Aspect: **Security**

Security /1

- ❑ Third Party Libraries safe?
z.B. neue Lib hat Vulnerabilities, ist deprecated/outdated, nicht vertrauenswürdig
- ❑ Authentication korrekt gesetzt?
z.B. neuer Endpunkt durch API-Token gesichert und autom. Tests dafür vorhanden
- ❑ Encryption genutzt und aktuell?
z.B. für Nutzer-Passwörter
- ❑ Code Secrets-frei?
z.B. Token klartext-gespeichert im Code/Config-File

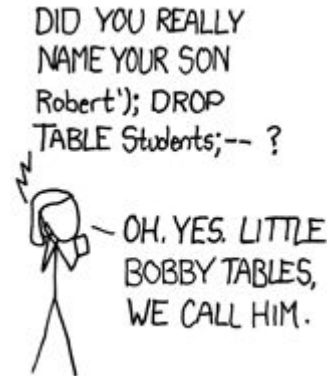


© refactoring.guru

Security /2

- ❑ Datenbank-Queries safe?
z.B. mögliche SQL-Injection weil keine SQL-Parameters genutzt
- ❑ Logging vorhanden und korrekt?
z.B. Rechteänderung in Logger aufgenommen und autom. Tests dafür vorhanden

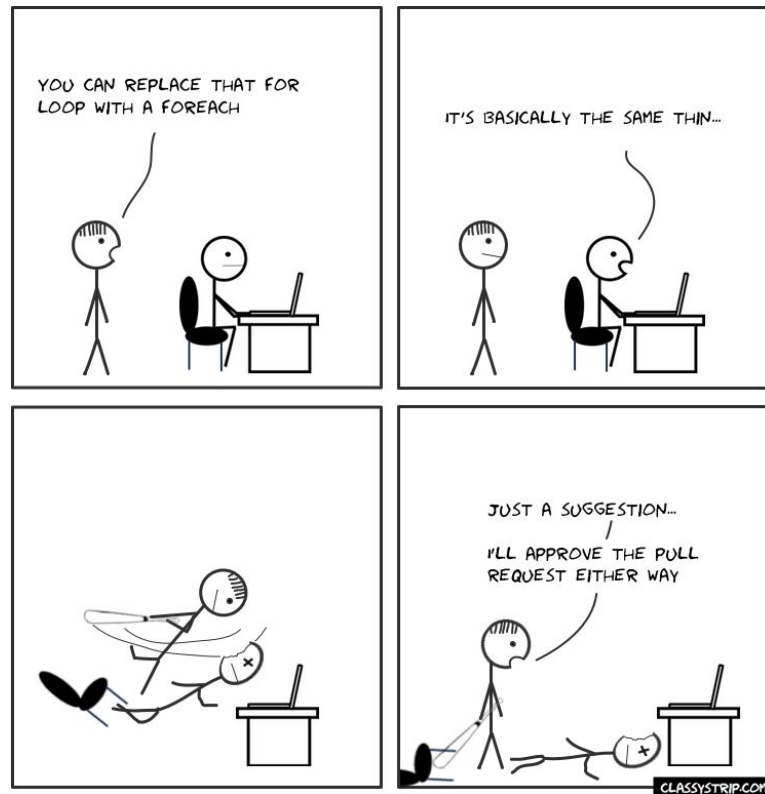
→ OWASP Top 10 - 2021:
<https://owasp.org/Top10/>



VII) Code of Conduct

Code of Conduct /1

- Trau dich
- Kritisiere sachlich statt persönlich
- Begründe deine Kritik
- Mach Verbesserungsvorschläge
- Unterscheide zwischen Nice-To-Have & No-Go



Code of Conduct /2

- ‚Not my Code‘: Verlange Boyscouting
- Sprich mit dem Autor
- Unterlasse Nit-Picking (automatisiere stattdessen)
- Gib Kudos
- Mach es täglich, mehrmals ;-)

PS: Sei dein erster Reviewer



Fazit

Fazit



Quelle: Office Space

**DAS KANNSTU
SCHON SO
MACHEN, ABER
DANN ISSES HALT**



KACKE

Quellen & weiterführende Links /1

1. The Upsource Blog: ***What to look for in a Code Review*** | Trisha Gee, 2015
blog.jetbrains.com/upsource/2015/07/23/what-to-look-for-in-a-code-review/ ff
eBook: leanpub.com/whattolookforinacodereview
2. The Upsource Blog: ***Code Review Best Practices*** | Trisha Gee, 2018
blog.jetbrains.com/upsource/2018/08/30/code-review-best-practices/
3. The Overflow: ***How to Make Good Code Reviews Better*** | Gergely Orosz, 2019
stackoverflow.blog/2019/09/30/how-to-make-good-code-reviews-better/
4. itemis: **10 Best Practices für Code-Reviews, die Spaß machen** | Sascha Bleidner, 2019
blogs.itemis.com/10-best-practices-fuer-code-reviews-die-spass-machen
5. ***How to Do Code Reviews Like a Human*** | Michael Lynch, 2017
mtlynch.io/human-code-reviews-1/
6. Atlassian: ***Why code reviews matter (and actually save time!)*** | Dan Radigan
atlassian.com/agile/software-development/code-reviews

Quellen & weiterführende Links /2

7. sd&m Conference: **Reviews and inspections / A History of Software Inspections** | Michael Fagan, 2001
mfagan.com/pdfs/software_pioneers.pdf
8. The Space Blog: **Best Code Review Tools for 2022 – Survey Results** | Evgenia Verbina, 2021
blog.jetbrains.com/space/2021/12/15/best-code-review-tools/
9. **OWASP Code Review Guide** | Larry Conklin and Gary Robinson
owasp.org/www-pdf-archive/OWASP_Code_Review_Guide_v2.pdf
10. Medium: **Code Reviews Code of Conduct** | Matti Bar-Zeev, 2018
medium.com/front-end-weekly/code-reviews-code-of-conduct-6c78a026ed35
11. **Characteristics of Useful Code Reviews: An Empirical Study at Microsoft** | A. Bosu, M. Greiler, C. Bird, 2015
miangshu.com/papers/CodeReview-MSR-2015.pdf
12. Heise: **Nachgebessert: Pull-Request-Workflows in der Praxis** | Jan Petzold, 2019
heise.de/ratgeber/Nachgebessert-Pull-Request-Workflows-in-der-Praxis-4341944.html