# IPC 2019: Aggressiv PHP QA

@Ocramius
@RoaveTeam

https://photos.app.goo.gl/Lc7wfrRdAmCf5pr1A

20 Reviews pro Tag
das ist trainiert, man sieht mehr Dinge als untrainierte

Talk: "Extremely defensive PHP" (2014)
http://ocramius.github.io/extremely-defensive-php/#/

Software Quality Assurance (Zusicherung)

speziell in Dtl wollen kleine Firmen wie große agieren -- mit Bürokratie "haben wir immer schon so gemacht"

Änderungen von Code in Produktion kostet viel mehr als vorher

http://www.agilemodeling.com/essays/costOfChange.htm
(DRUCKEN! Figure 3. Comparing the feedback cycle of various development techniques.)

nicht Testen ist nicht professionell :)
manuelle Tests sind Bullshit

"traditionelles" Software Testing ist TOT
Dev ist selbst QA

was hilft: BDD, DDD, Emerging Specifications

## Architecture

Non-functional requirements

Dokumentation via ADR direkt im Repo

(TODO ADR: Value-Objekte sind Immutable)
(TODO ADRs zu allen relevanten PHP7 Paketen)

CI: all_good() || exit(1)

## Tools

https://github.com/sensiolabs-de/deptrac
Erkennt wenn Architektur gemixt wird, z.B. Repository wird von Repository verwendet, nicht ausschließlich von Services
(TODO: in CI integrieren)

Opensource: Psalm, PHPStan
Closed: Exakat, Blackfire
Ocramius bevorzugt Psalm
Ocramius installiert keine Extensions :), Phan bräuchte Extensions

Ohne Type System
$apples + $oranges :)

Ocramius installiert Tools tatsächlich in composer.json --dev und nicht via PHARs
wg. Updatechecker und vermutlich Exploit-Finder

Types etc sind nicht teil des Reviews

unbedingt verwenden!

```
/** psalm-immutable */
class Bla { public $foo; }
```

Property kann nicht überschrieben werden

Psalm schaut welche Typen er interpretieren kann (infer types) - 99% ist gut, 80% ist ein Smell
kann er nicht: json_decode, eval
dann muss man Psalm via Annotations lehren wie er es zu interpretieren hat

Psalm unterstützt Baseline: "--set-baseline" bekommt eine Liste mit zu ignorierenden Dingen
gut für Legacy Systeme die nicht noch schlechter werden sollen

Psalm unterstützt Generics
`/** return list<int> */`

Mit Types kann eine ganze Kaskade an Typprüfungen im Code und Tests weggeworfen werden

https://github.com/Roave/you-are-using-it-wrong
wird von Package Autoren verwendet
wenn das Paket installiert wird, wird der Projekt-Code auch mit Static Analyser geprüft
wenn das fehlschlägt, darf man das Paket nicht installieren :)

Mutation Tests lassen nicht die Static Analysis vorher laufen
macht er aus `array_values(array_filter(..))` ein `array_filter(..)` fischt einen das die SA raus, braucht man keinen Test für schreiben

https://github.com/doctrine/coding-standard

## Benchmark

https://github.com/phpbench/phpbench

Tideways, Sentry, Instana, Datadog

## Update Dependencies

roave/backward-compatibility-check

Dependabot für Github

roave/security-advisories

maglnet/composer-require-checker

icanhazstrings/composer-unused