Das Gesetz der 2 Füße!

Es beginnt & endet, wenn die Zeit reif ist!

Was auch immer geschieht, es ist das Einzige, was geschehen konnte!

Wer auch immer kommt, es sind die richtigen Leute!

#xdde23

andrena Experts in agile software engineering

1

Teilzeit schlägt Vollzeit

+ homogene Teilzeit
  „alle die gleichen Tage frei"

+ Fehltage/-zeiten einplanen
  ↳ + Leute „abholen" ⇒ Planbarkeit ist wichtig!
  ↳ Pull & Push-Prinzip

+ Gut eingespielte Teams
  ↳ In Teambuilding investieren

+ Für Schnittmengen sorgen

− Remote verschärft die Situation
  ↳ Mittagessen, Kaffeepause zusammen fehlt

+ Infos „festhalten", z. B. leichtgewichtig in Channel posten

(2)

Teilzeit schlägt Vollzeit

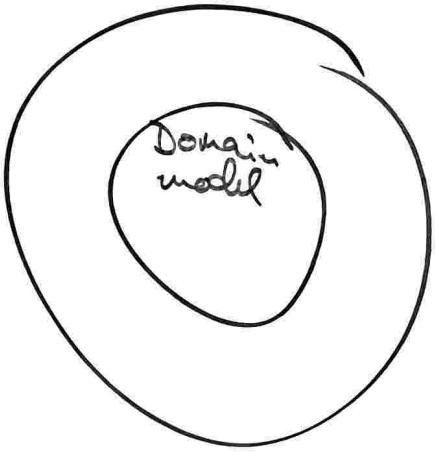
+ Fokus
  - Möglichst nur ein Projekt
+ Pair Programming für Knowhow-Transfer
+ Stories klein genug schneiden & einplanen

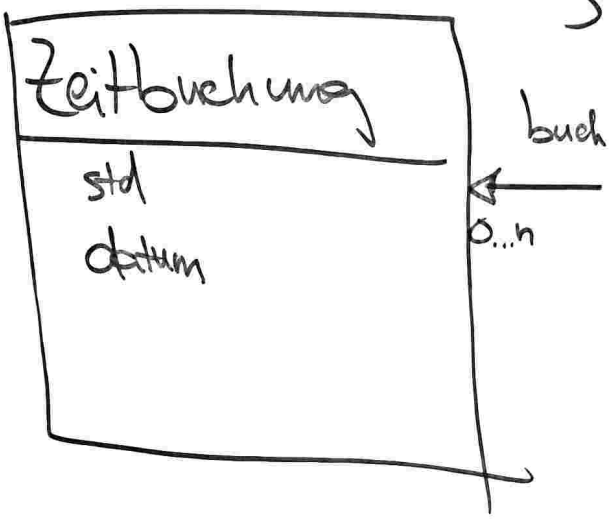
OO in the wild?!

- Primitives kapseln in Fachobjekte
- CRUD nur wenn es keine Beziehungen gibt
- Beziehungen → Validierungen

REST

{ std: 8
auftrag: 4
mitarbeiter: 002
datum: 2023-10-06 }





Tagesbuchung
kann entscheiden ob Zeitbuchung valide ist

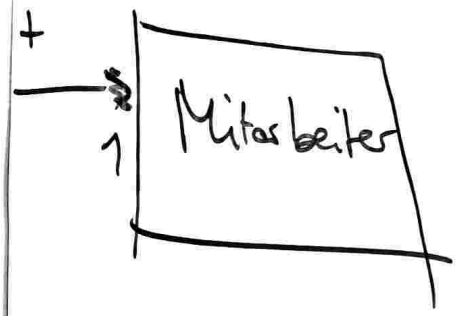
OO → Objekte "erfinden"
↖ Name!

wenn es eine Zuständigkeit gibt, die keine Heimat hat

↳ CRC

↳ daraus sollte sich ein guter Name ergeben

- Objekte mit Leben füllen



DevOps

- 1 Ops / Team
- Fullstack, auch Betrieb
  jede* Entwickler*in – ist es noch effektiv? qualitativ hoch?
  = jeder kann alles, aber nichts richtig gut
  ⇝ besser: Team ist Fullstack
- jeder hat seinen Schwerpunkt, aber jeder kann releasen
- Plattformteam befähigt Dev Team
- Supportteam bringt Ops Knowhow in die Teams
- Je nach Komplexität der Technologien
  ↳ s. Team Topologies Grenzen Größe
- Gildenkonzepte CoP
- Entscheidung sollte Teil der Selbst-orga. sein!

andrena Experts in agile software engineering

37 Arten

Funktionales Verhalten

- A/B Test
- UX Test
- UI Test
- Unit Test
- Integration Test
- E2E Test
- Fuzzing Test (Hypothesis tests)
- Regressions test
- ~~Mutation Testing~~
- Screenshot Test
- Charakterisierungs test
- ~~Akzeptanz~~/Progressionstest
- Akzeptanz test
- Explorativ
- Barrierefreiheit
- Contract Test
- Smoke-Test

Performance

- Last test
- Stress test (until failure)
- Leistungs test
- Scalability

Interne Qualität

- Mutation Coverage
- Line Coverage
- Condition Coverage
- Statement Coverage
- Linting
- Code Smell Analyse
- Compiler Warnings
- ArchUnit

Security

- Statische Dependency-Analyse
- Pen-Test (lab-automatisiert)
- Statischer Bad Practices Scan
- Ausfallsicherheit (Chaos Monkey)
- White/Gray/Black-Box

AI for Coding

Copilot: Tests generieren
Code Completion
Versionen hochziehen

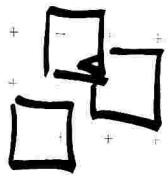
ChatGPT: Erklärt Code
Refactoring

Lizenzfrage für generierten Code?

andrena Experts in agile software engineering

Visualisieren

Mit/ohne UML

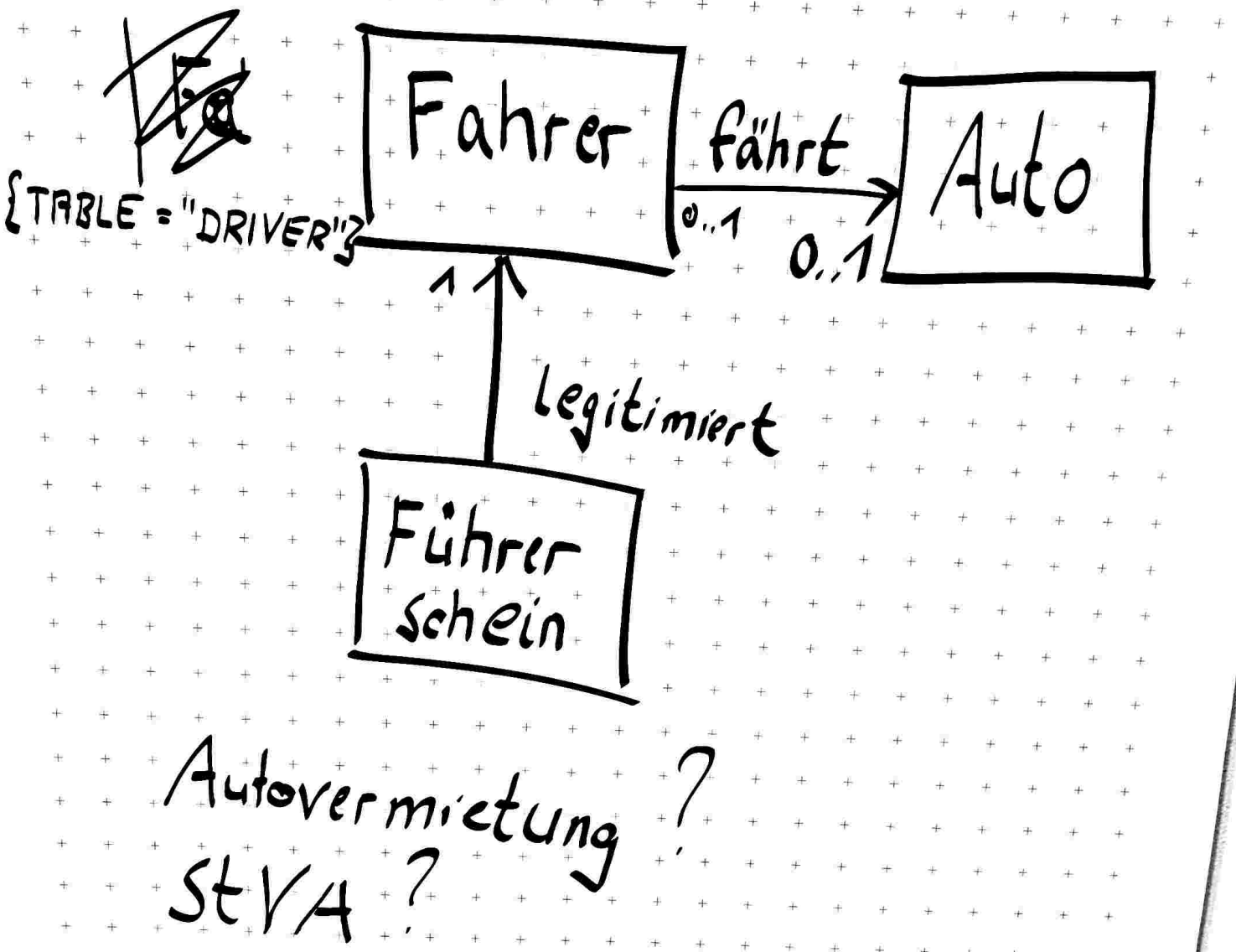


„Bunte Bilder"
reichen (erstmal)

keine Motivation
zur Pflege der Diagramme

Alternativen

Draw IO



Autovermietung ?

StVA ?

Agile Craftmanship – brauchen wir so was?

→ hinkt der Vergleich zu Software Craftmanship?

z.B. Katas (Fallberatung)
Rollenspiele
Samanan Coaching
Kollegiale Lernbegleitung
Regelmäßig Üben

Moderation
Konfliktmgmt...

HINDERNISSE

- Selbstüberschätzung & faul
- Agile Rollen sind wenig ausdefiniert?
- Sind die Ziele und Mehrwert bekannt?
- Wissen vs Können
- Ich habe keine Zeit!
- Unterschiedliche Lerntypen
- Verantwortung nicht klar
- Ich muß mit meinen eigenen Fehlern/Schwächen umgehen
- Lernpartner, dem ich vertraue

Wie gehen wir mit neuem Wissen um?

Wie gehen wir mit neuem Wissen um?

- Austausch
- Case Studies, Case Clinic

Theorie U

- Was passiert, wenn wir nicht vorankommen?

- Oft keine Ursache-Wirkungs-Ketten

Wieviel Softwarearchitektur ist notwendig?

Accidental Architecture → Artikel von Grady Booch

zufällig, verunfallt
- so viel bisher (nicht entscheiden)
- Konferenz-getriebene Entscheidungen

nicht dokumentiert
↓
ggf. Kopf-Wissen

explizit
- Frage stellen: "Was sind unsere technischen Anforderungen / Ziele
  - ↳ daraus Entscheidungen bewusst ableiten

dokumentiert → arc42

Ressourcen:
- Adam Bien (Microservices in Java EE8) → Youtube
  - ↳ "Wenn ihr eine verteilte Architektur braucht → dann werdet ihr es merken!"
- Peter Fichtner (Vortrag beim Entwicklertag Karlsruhe): "Wenn Microservices die Antwort ist, was war eigentlich die Frage?"
  - ↳ Youtube

Observability

- Monitoring als Bestandteil des Daily
- Die Vorstellung der Dashboards im Daily 'rotiert'. Jede(r) ist mal dran
- Monitoring ist für das TEAM!
- Stress-Level (DevOps) sinkt
- Umgang / Wissensverbreitung verbessert

SOLID vs. CUPID

↓ code-näher restriktiver, Domain agnostisch

? Behauptung: SOLID nicht nicht?

↓ abstrakter, offener

Dan North ist Autor (weniger dogmatisch)

- erst der Name, dann die Prinzipien!

Video von Kevlin Henney: SOLID (2013)

- SOLID ist hilfreich
- löst aber nicht alle Probleme
- teilweise zu dogmatisch
- erst die Prinzipien
  ↳ Michael Feathers hat den Namen geprägt

Composable

↳ Unix-Philosophie

arbeiten gut zusammen:

- Predictable
- Idiomatic
- Domain-based

- Artikel von Robert C. Martin (2000): "Design Principles and Design Patterns"
  ↳ viel mehr Prinzipien (SRP fehlt?)