# Projektabschluss

Team Dominic, Sebastian, Jonas, Solaiman

# Teamarbeit

- schwierig per Remote
- Verantwortung trägt jeder!
- Aufgabenverteilung:
  - konnte teilweise nicht parallelisiert werden
  - kleine Code Base führte zu Abhängigkeiten in der Implementation
  - -> gute Absprachen

# Legacy Software

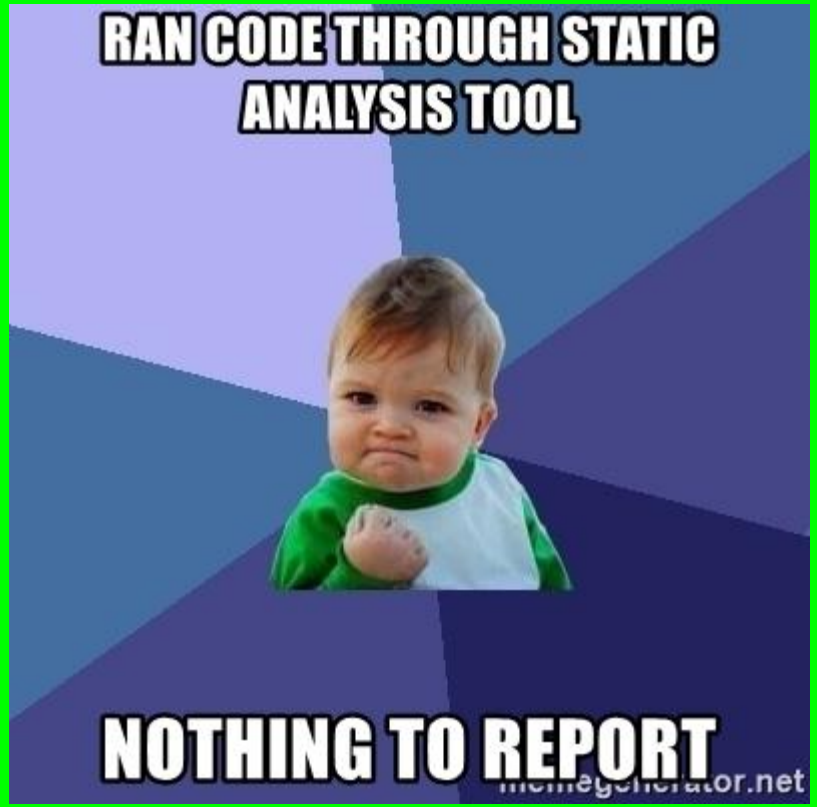
- Verständnis entwickeln ist nicht trivial
- Abwägungen zwischen neuen oder bisherigen Funktionalitäten
    - Wrapper?
- nicht immer erkennbar, was noch wichtig ist
- experimentelles Debuggen

# Dokumentation und Kommentare

- hilfreich für andere:
  - fehlte im Legacy Code
  - bei Weiterentwicklung
  - bei Nutzung (STRG+Q)
- nervig für die Entwicklung
  - Getter/Setter?!
  - Konstruktoren?!
  - auto-generierter Stuff
- nachträgliche Dokumentation noch nerviger

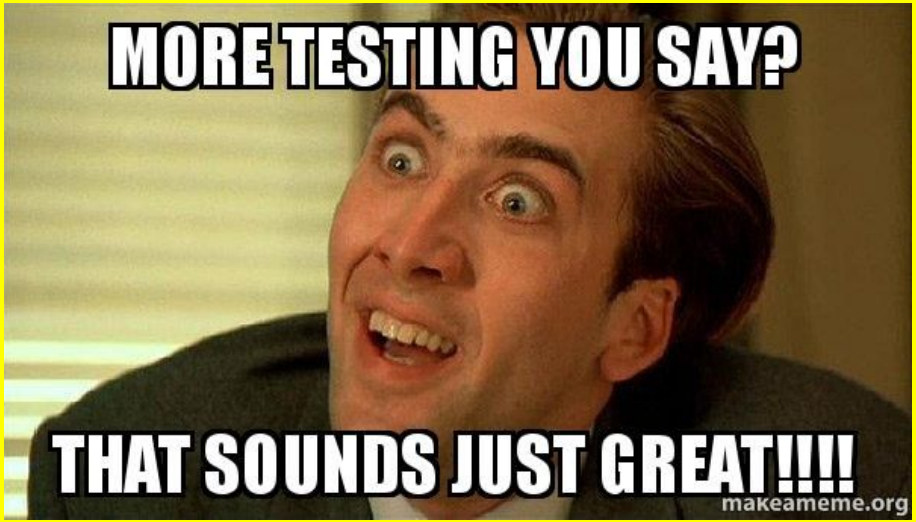
# Statische Code Analyse

- kann zeigen, ob Code fehleranfällig ist
  - Endlosschleifen
  - komplizierte boolean Ausdrücke
- Metrik muss unbedingt im Team abgesprochen und priorisiert werden
- Funktionen werden unleserlicher, wenn sie um der Metrik Willens “refactored” werden



# Entwicklung von Tests

- Testfälle finden und definieren erfordert “Thinking out of the Box”
- Schwierig bestimmte Testfälle anzusteuern
  - std::cin
  - void-Funktionen
  - Umgebung aufbauen
    - Instanzen erstellen
  - Mocks und Captors gehören zur Test-Toolbox
- schlechtes Design wird ersichtlich



# Test Coverage

- “Fake it, till you make it”
- Neue Funktionalität, massive Veränderungen, aber kein Test geht kaputt?!
- extreme Edge-Cases nicht zu testen gewesen
- Metrik war sehr stupide

# Schätzungen

- brauchen viel Team-Erfahrung
  - muss wachsen im Team
- meistens dauert die Umsetzung länger als gedacht
- Komplexität statt Zeit ist wesentlich hilfreicher
- Deadline festlegen extrem schwierig
  - Velocity entwickelt sich erst

# Konfigurations-management

- Verknüpfung der IDE mit Jenkins
  - Jenkins lieferte mehr Ergebnisse zurück
  - IDE lieferte nicht alle Fehlermeldungen / Ergebnisse
- Gute Erfahrungen mit der Versionsverwaltung (git)
  - Nachvollziehbar wer, welche Änderungen vorgenommen hat
  - “Back-Up’s” waren vorhanden

# Unsere “Definition of Done”

- Funktionalität
- Metriken müssen passen

**Warum wichtig?**

- erkennen, ob man fertig ist
  - abhaken
- “... weil der Prof sonst meckert”
- Kann weitergearbeitet werden?
  - Serialisierung der Aufgaben
- schlechte Definition -> Kunde unglücklich, Geld weg, neu anfangen

**Wer hat sie definiert?**

- Team (20-45%)
- Praktikumsleiter (80-55%)

# Qualität ist eine Strafe für Entwickler ! ?

- Schwierig die Metriken auf einen bestimmten Schwellwert zu kriegen
  - Metrik != Qualität
- Funktionen müssen kommentiert werden die eigentlich verständlich und lesbar sind
- Schwierig Qualität aufzubauen
  - Schwieriger ohne Qualität zu arbeiten

Was war gut im Praktikum? Was war schlecht?

# Wodurch unterscheidet es sich von anderen Praktika im Studium?

Was bleibt dauerhaft in Erinnerung?

Was würden Sie selbst anders machen, wenn Sie das Praktikum nochmal machen müssten?