Notes

Titelslide

Welkom allemaal, ik ga jullie wat meer vertellen over het optimaliseren van continuous integration door middel van test case prioritization

Overview

- Eerst leg ik uit wat CI is.
- Dan leg ik uit welk probleem ik wil oplossen.
- Vervolgens leg ik uit welke oplossingen er bestaan.
- Wat ik al gedaan heb en wat ik nog ga doen.

CI

Wat is CI nu precies? Ik ga dit uitleggen aan de hand van een voorbeeld.

- Beschouw de ontwikkeling van een Android applicatie.
- Verschillende developers van de applicatie werken elk aan de code op hun eigen pc.
- Van tijd tot tijd pushen ze deze code naar een gemeenschappelijke server.
- Op die server worden een aantal tests uitgevoerd.
- Die al dan niet kunnen slagen of falen.
- Wanneer alle tests slagen wordt de app naar productie gedeployd, in dit voorbeeld dus naar de playstore.

Problem

- Nu, wat is het probleem precies?
- Die tests zijn het probleem

Tests

- Daarnet waren er weinig tests -> alles oké
- Naarmate software groeit
- Worden dat plots veel tests die heel lang duren om uit te voeren

Existing solutions

Nu, wat kunnen we daaraan doen? 3 mogelijke strategieeen

Test Case Selection

TCS gaan toepassen. Gegeven alle tests, pik er enkele uit en voer al de rest simpelweg niet uit.

Test Suite Minimisation

TSM. Gelijkaardig aan TCS, maar niet at runtime. Eerder een techniek die je op voorhand kan toepassen op een set tests om die permanent weg te gooien.

Test Case Prioritization

Alle tests uitvoeren, maar in een zodanige volgorde, dat de tests waarvan we denken dat ze gaan falen, als eerste worden uitgevoerd. Op die manier kan de totale test sequence sneller worden afgebroken als er al één faalt en kan je al beginnen met bugfixen.

- Kijk naar voorgaande uitvoeringen van die test
- Kijk naar de test coverage: -> Welk stuk code heeft invloed op welke test? Doel is om met zo weinig mogelijk tests (zoals in TCS) zo snel mogelijk, zoveel mogelijk code te coveren
- Tunen:
  - branch of statement coverage?
  - Om zo weinig mogelijk tests te hebben moet je definiëren hoe 2 tests van elkaar verschillen via een afstandsfunctie.

Roadmap

Parsing

- Projecten gezocht
- Parsers geschreven voor test runs te analyseren; Github Actions en Travis CI
- Naar JSON

Literature study

- Literatuur studie aan het uitschrijven, 16 bladzijden

Implementatie

- Verschillende TCP algoritmes vergelijken, eventueel zelf nog zoeken.
- Metapredictor, verschillende algoritmes uitvoeren en kijken welk algoritme het beste werkt voor welke codebase en daar dan meer belang aan hechten.
- Plugin voor Jenkins schrijven.

Quote

Tot slot nog deze quote van Benjamin Franklin, wat de motivatie voor deze thesis vormt: tijdswinst.

Zijn er nog vragen?