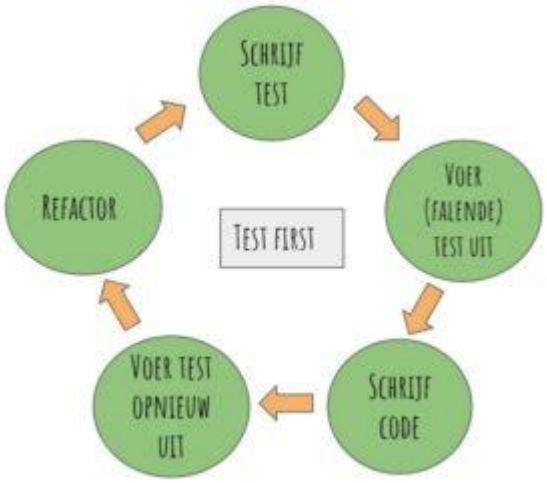
Software Testing

Test-driven development



Bij het ontwikkelen van software wordt vaak eerst de code geschreven en daarna wordt er getest. Dat klinkt heel logisch. Want er valt niets te testen als er geen code is, toch?

Een van de grootste uitdagingen bij deze manier van ontwikkelen is dat het testen pas laat in het proces wordt uitgevoerd en blijkt het ook moeilijk te zijn om een goede testdekking te realiseren.

Toen ergens in de jaren ‘90 Extreme Programming (XP) werd geïntroduceerd werd daarbij ook test-first geïntroduceerd. En daar wordt echt mee bedoeld dat er eerst testen worden gemaakt en uitgevoerd en dat pas daarna de code wordt geschreven. De testcases worden opgesteld in een samenwerking tussen business, testers en ontwikkelaars. Het helpt ontwikkelaars software te schrijven die aan de testcases voldoet.

Op enig moment kwam, uit Scrum teams, de volgende vraag naar boven: Wat als we een manier zouden kunnen vinden om de voordelen van “test first” development ook te kunnen gebruiken voor functionele testen en acceptatietesten? Toen ontstond Acceptance Test Driven Development (ATDD). Later wilde Dan North het gedrag benadrukken vanuit een business perspectief en gaf zijn aanpak de naam Behaviour Driven Development (BDD). ATDD en BDD zijn in de praktijk vergelijkbare aanpakken. De verschillende test first aanpakken worden ook wel aangeduid met X-driven development, waarbij de X staat voor de drijvende kracht. Bij TDD is dat de unittest, bij ATDD is dat de acceptatietest en bij BDD is dat het gedrag dat de gebruiker van de software verwacht.

Definitie Test Driven development (TDD)

Test Driven Development is een programmeer stijl waarbij drie activiteiten nauw met elkaar zijn verweven: codering, testen (in de vorm van unit test) en ontwerp (in de vorm van refactoring)

Definitie van wiki:

***"Test-Driven Development** (TDD) is een ontwikkelmethode voor software waarbij eerst tests worden geschreven en daarna pas de code. De naam Test-Driven Development komt van Kent Beck, die deze techniek in 2002 op papier heeft gezet en daardoor de bekendheid ervan verbeterd heeft. Het valt onder de agile-softwareontwikkeling."*

TDD is een methode waarbij, in kleine incrementele stappen, unit tests worden gemaakt, en in dezelfde kleine incrementele stappen wordt de software ontwikkelt om aan die tests te voldoen.

De ontwikkelaar hanteert daarbij de volgende 5 stappen:

1. schrijf een unit test die een deel van de software test
2. voer de test uit, deze moet mislukken omdat in de code de functionaliteit nog ontbreekt
3. schrijf "net genoeg" code, om de test te laten slagen
4. voer de unit test opnieuw uit
5. als de test slaagt, ga dan verder met de volgende test, en anders herschrijf / wijzig de software om de test te laten slagen

Het resultaat is een volledig geautomatiseerde reeks unit tests. De unit tests zijn echter niet het primaire doel van TDD. Het gaat veel meer om het van te voren bepalen van concrete en gedetailleerde verwachtingen over het gedrag van de software, dan dat het gaat om testen.

Ontwikkelcyclus

De ontwikkelcyclus hieronder is gebaseerd op het boek *Test-driven development by example* van Kent Beck. Deze cyclus kan zo vaak herhaald worden als nodig is om de code volledig werkend te krijgen volgens de requirements.

1. **Test maken:** In Test-Driven Development maakt de programmeur eerst een test, gebaseerd op een requirement, en schrijft pas dan de code.
2. **Alle tests draaien en kijken of de nieuwe test faalt:** Deze test moet initieel in principe falen, aangezien het stuk code waarop deze test van toepassing is nog niet bestaat. Het laten falen van de nieuwe test is belangrijk om na te gaan of de test effectief is (geen fouten bevat en daardoor bijvoorbeeld altijd slaagt).
3. **Code schrijven:** In deze stap wordt de daadwerkelijke code geschreven om de zojuist gemaakte test te laten slagen. Deze code mag alleen functionaliteit bevatten die nodig is om de test te laten slagen. De code hoeft niet perfect geschreven te zijn, zolang deze maar werkt. In een latere stap zal de code nog herschreven worden volgens de standaard.
4. **Tests draaien en kijken of deze slagen:** In deze stap draaien alle tests nogmaals en wordt gekeken of alle tests slagen. Is dit het geval dan is dit een goed uitgangspunt om aan de volgende stap te beginnen.

5. **Code herschrijven:** In deze stap wordt de code verfraaid en volgens de standaard herschreven. Ook zal eventuele dubbele code, nodig om de onderlinge afhankelijkheid van de componenten te minimaliseren, verwijderd worden.

Voordelen

- Er wordt bij Test-Driven Development gekeken vanuit het perspectief van de gebruiker. De testcases waarvoor de code wordt geschreven zullen namelijk gebaseerd zijn op de requirements, die vanuit het oogpunt van de gebruiker zijn opgesteld. Problemen met de bruikbaarheid van de interface kunnen zo eerder worden gevonden en gecorrigeerd in een stadium waarin het gemakkelijkst is
- Er wordt specifiek gelet op de requirements, aangezien de tests daarop gebaseerd zijn. Extra, wellicht overbodige, functionaliteiten zullen hierdoor niet geïmplementeerd worden, waardoor de hoeveelheid programmacode binnen de perken blijft.
- Doordat alle code van het begin af aan wordt getest, wekt dit meer vertrouwen bij het ontwikkelteam en bij de klant.
- Ondanks de extra code die nodig is voor het schrijven van de tests, zal de ontwikkelingstijd toch korter worden, aangezien fouten al in een zeer vroeg stadium gevonden zullen worden.
- Aangezien alle onderdelen van de code los van elkaar getest worden is de onderlinge afhankelijkheid kleiner en daarom minder complex.

Nadelen

De programmeur schrijft bij deze methode zowel de tests als de code voor de applicatie. Dit heeft tot gevolg dat wanneer de programmeur iets over het hoofd ziet, dit in zowel de test als in de code over het hoofd gezien zal worden.

- Wanneer er een groot aantal tests succesvol is, kan dit de indruk geven dat de applicatie volledig getest is. Een valkuil is dat een integratie- en systeemtest niet of nauwelijks uitgevoerd worden.

Toepassing

Deze methode maakt gebruik van korte ontwikkelcycli met het continu testen van de software. Voor het ontwikkelen/upgraden van toolkits, is Test-Driven Development een toepasbare manier van ontwikkeling, aangezien integratie- en systeemtests hierbij nog niet ter zake doen.

Bronnen

- TestDrivenDevelopment op WikiWikiWeb
- Test or spec? Test and spec? Test from spec!, by Bertrand Meyer (september 2004)
- Microsoft Visual Studio Team Test from a TDD approach
- Write Maintainable Unit Tests That Will Save You Time And Tears
- Improving Application Quality Using Test-Driven Development (TDD)
- https://www.youtube.com/watch?time_continue=16&v=W-I1pfNmAy0&feature=emb_logo