Development

Behavior Driven development (kortweg BDD) is geen nieuw framework of een compleet nieuwe methodiek voor software development. Het is een collectie van bestaande technieken die gericht zijn op het opstellen van duidelijke testcases die door zowel de developers als de gebruikers wordt begrepen.

Behavior Driven Development

BDD uitgelegd met voorbeelden van EasyB

Ronald Haring
senior ontwikkelaar iProfs.

BDD is Test Driven Development maar dan beter uitgevoerd. Om te begrijpen waarom BDD beter uitgevoerde TDD is, moeten we eerst inzichtelijk maken waar TDD zich op richt. TDD houdt zich bezig met het unit testen van een systeem. Een unit is in dit geval een onderdeel van een systeem, zoals class of een methode van een class, en deze methode of class wordt vervolgens getest. Met als gevolg dat de structuur en denkwijze van de testcases gelijk lopen aan de structuur en denkwijze van de classes of methodes.
Een test zou zich niet moeten richten op een technisch aspect (zoals een class of een methode, of hoe de classes en methodes elkaar aanroepen) maar op het gedrag van het systeem. In plaats van testen te schrijven die controleren wat de code doet, zouden de tests moeten beschrijven of specificeren wat het systeem zou moeten doen. Het gedrag van het systeem in bepaalde situaties moet getest worden.
Dit komt ook dichterbij een complete TDD aanpak waar begonnen wordt met de testcase te schrijven en daarna pas de code. Het schrijven van een testcase is echter een stuk makkelijker als er niet van te voren vastgelegd hoeft te worden welke class of methode iets gaat uitvoeren, maar als je het gedrag wat je verwacht beschrijft in een testcase. Vervolgens kun je aan de hand van dit gedrag de classes en methodes gaan invullen.

Gemeenschappelijke taal

Om het gedrag van het systeem te kunnen beschrijven is het belangrijk dat de gebruikers van het systeem erg betrokken zijn en helpen bij het beschrijven van de specificaties van het systeem. De meeste gebruikers zijn hier niet ervaren in. Dit levert vaak onduidelijke en onvolledige requirements op en dus later in het traject problemen. Om te zorgen dat de specificaties zoveel als mogelijk aansluiten bij de ideeën van de gebruikers, dienen alle betrokkenen dezelfde woorden te gebruiken. Voor elke betrokkene dient er een duidelijke afbakening te zijn wat de gebruikers nu precies bedoelen met hun woorden. Deze ideeën van de gebruikers worden vervolgens als een gemeenschappelijke taal gehanteerd die door het hele team gebruikt zullen worden. Als een gebruiker het vervolgens heeft over een bepaald concept (zoals bijvoorbeeld een debiteur), dan dient iedereen te weten wat er met een debiteur bedoeld wordt.

User story

De gemeenschappelijke taal (een concept afkomstig uit de domain driven development hoek) wordt vervolgens gebruikt om voorbeeld specificaties te beschrijven die door de eindgebruikers begrepen kunnen worden. Deze specificaties worden op een dusdanige manier vastgelegd dat de eindgebruikers deze ook makkelijk kunnen lezen. De gemeenschappelijke taal kan opgesteld zijn in gewone text bestanden die later gekoppeld kunnen worden aan een programmeurstaal (zoals JBehave met Java) of in een DSL die gemaakt is voor het opstellen van de testcases in de gemeenschappelijke taal (zoals EasyB in Groovy)

Een testcase voor BDD komt over het algemeen overeen met het opstellen van een user story. Dit kan gebruikt worden om globale specificaties op te schrijven zoals:

- Als een persoon in een bepaalde rol
- Wil ik dat het systeem het volgende doet
- Zodat er een bepaalde opbrengst is

Indien een user story op deze manier wordt opgesteld, wordt meteen duidelijk wat de meerwaarde is van het systeem voor bepaalde personen.
Om vervolgens vast te stellen wat het systeem doet binnen specifieke situaties, dienen de acceptatie criteria getest te worden. Deze acceptatie criteria (of specificaties van het systeem) kunnen dan als volgt opgesteld worden:

- Gegeven een bepaalde situatie van het systeem
- Als ik in dan bepaalde acties uitvoer
- Dan zou het volgende zich voor moeten doen.

Een enkele specifieke situatie van een user story wordt een scenario genoemd. Een user story kan opgebouwd zijn uit een aantal scenarios die allemaal een ander aspect van de user story kunnen belichten.

Indien de testcases op deze manier opgesteld worden, dan kunnen de gebruikers van het systeem beter aangeven wat het systeem nu zou moeten doen in die situaties. De gebruikers helpen dan actief mee om hun systeem te bouwen aangezien ze zelf kunnen lezen wat het systeem gaat doen.
Een alternatieve manier voor het opstellen van gedrag is door gebruik te maken van specificaties.

Specificaties

Een specificatie staat wat meer los van de user stories. Bij een user story wordt uitgegaan van een bepaalde gedrag wat in een bepaalde situatie zich voor dient te doen, terwijl bij een specificatie driven framework het gedrag wordt gespecificeerd en dit wordt getest. De specificatie staan iets dichter bij unit testen dan de user stories. Het leest alleen wel wat makkelijker dan een unit test

Een voorbeeld van een specificatie zou kunnen zijn:

Voor een nieuw aangemaakte gebruiker op de site geldt:
- dat de gebruiker kan inloggen
- dat de gebruiker berichten kan posten in het forum
- dat de gebruiker zijn wachtwoord kan aanpassen

Een scenario voor een user story hiervoor zou kunnen zijn
- Gegeven het feit dat een gebruiker zich geregistreerd heeft op de site
- Wanneer de gebruiker is ingelogd
- Dan kan de gebruiker berichten plaatsen in het forum
- En kan de gebruiker zijn wachtwoord aanpassen

BDD Voorbeeld

Voor het opstellen van een voorbeeld situatie ben ik uitgegaan van een aantal simpele user stories die te maken hebben met een website waarop mensen zich kunnen aanmelden bij verschillende communities en waarop zij berichten kunnen plaatsen in een forum.

User story:
- Als een gewone gebruiker
- wil ik mij aan kunnen melden bij een bepaalde community met een username en wachtwoord
- zodat ik kan inloggen en deel kan nemen aan deze community

Scenario: een nieuwe gebruiker wordt lid van een bestaande community
- Gegeven een bestaande community “test”
- Als een niet bestaande gebruiker met login “login” en wachtwoord “wachtwoord” zich registreert
- Dan moet de nieuwe gebruiker login kunnen inloggen met wachtwoord “wachtwoord”

Scenario: een nieuwe gebruiker wil lid worden van een bestaande community met een al bestaande login
- Gegeven een bestaande community “test”
- Als een gebruiker met login “login” en wachtwoord “wachtwoordNieuw” zich aanmeldt
- En er bestaat al een gebruiker met login “login”
- Dan moet de nieuwe gebruiker een melding krijgen
- En de nieuwe gebruiker kan niet inloggen met login “login” en wachtwoord “wachtwoordNieuw”

Scenario: een bestaande gebruiker kan inloggen
- Gegeven een bestaande community “test”
- En een bestaande gebruiker met login “login” en wachtwoord “wachtwoord”
- Indien deze gebruiker met login “login” en wachtwoord “wachtwoord” inlogt
- Dan moet de gebruiker ingelogd zijn

Scenario: een bestaande gebruiker probeert met verkeerde wachtwoord in te loggen
- Gegeven een bestaande community “test”
- En een bestaande gebruiker met login “login” en wachtwoord “wachtwoord”
- Indien deze gebruiker met login “login” en wachtwoord “wachtwoordTypo” inlogt
- Dan moet de gebruiker niet ingelogd zijn

Frameworks

Er zijn veel frameworks beschikbaar voor BDD. Voor Java zijn de bekendste JBehave en JDave. JBehave richt zich op de user stories en uitwerkingen hiervan, terwijl JDave zich richt op het uitwerken van

Er zijn veel frameworks beschikbaar voor BDD.