De manier van ontwikkelen verandert, maar het doel niet…

Nog altijd moet er software opgeleverd worden met voldoende kwaliteit. Welke impact hebben de wijzigingen in de ontwikkelmethode op het testen en hoe gaan we daar mee om? Wat verandert in testen? De impact is grofweg onder te verdelen in drie groepen:

1. Mens: communicatie en samenwerken
2. Proces: wat en wanneer te testen
3. Techniek: hoe te testen

1. Mens

Door de jaren heen moeten we steeds meer samenwerken in plaats van alleen maar samen werken. De mate van samenwerking neemt toe met de verschillende manieren waarop ontwikkeld wordt. Waar we bij Agile al meer moesten samenwerken met ontwikkelaars en businessanalisten, moet er bij DevOps ook nauwer samengewerkt worden met operations (beheer) en de gebruikers van het systeem in productie.

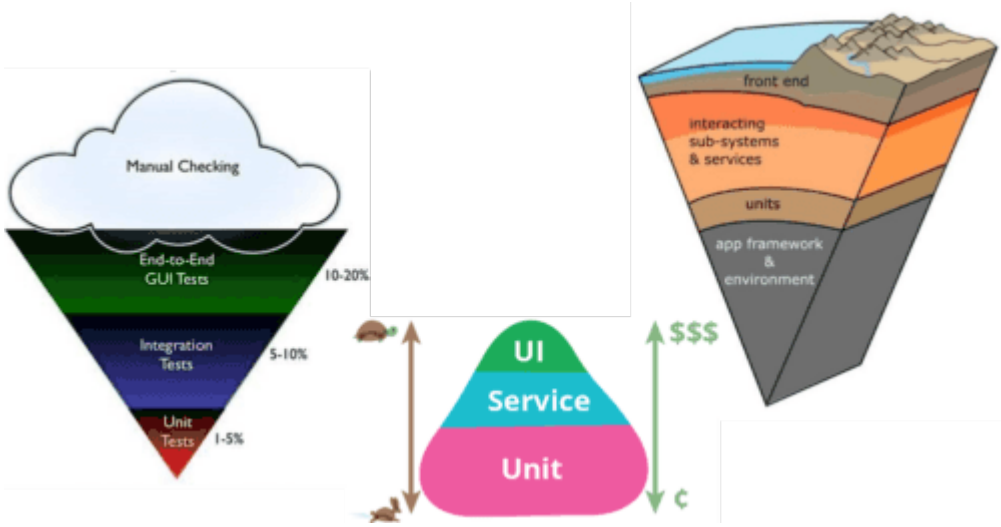
Naast het samenwerken wordt er ook steeds meer van mensen in de teams verwacht. Iedereen moet hetzelfde zijn: een DevOps engineer. Door management wordt dan vaak verwacht dat je alles kunt, zowel testen als ontwikkelen als ontwerpen als…. Maar is dit in de praktijk ook zo? We zien inderdaad generalisten die op alle verschillende vakgebieden kunnen ondersteunen en helpen, maar als er specifieke informatie nodig is over bijvoorbeeld testen, dan wordt toch snel naar een specialist op dat gebied gezocht. Hoewel iedereen gelijk is qua titel, zijn alle personen in de teams toch verschillend.

Een ander aspect is dat mensen die fouten durven maken, vaak beter gewaardeerd worden. Neem initiatief en probeer nieuwe aanpakken of technieken. Wanneer dit misgaat kun je altijd nog sorry zeggen, maar als het goed gaat heb je een zeer waardevolle bijdrage geleverd.

2. Proces

Goed nadenken over het testproces is noodzakelijk. Er moet vaker en sneller opgeleverd gaan worden, maar deze opleveringen moeten natuurlijk wel van voldoende kwaliteit zijn. Wat we vaak zien is de zogenaamde “ice-cone”, veel tests via de ui en steeds minder tests op de lagere niveaus van de applicatie. Hier is al redelijke tijd van bekend dat het niet de beste manier is om de zaken aan te pakken. De testpiramide geeft aan dat tests op de hogere niveaus niet alleen trager zijn dan de tests op lagere niveaus, maar ook nog eens meer kosten om te onderhouden. De recentste toevoeging aan deze modellen is het “round

earth model”. Hier wordt vooral naar risico’s gekeken. Wanneer een probleem zich manifesteert in de laag of lagen die door gebruikers worden afgenomen, is er een hoger risico. Wanneer we de laatste twee modellen combineren, komen we in de situatie dat we naar de juiste dekking op het juiste niveau zoeken, waarbij we goed kijken naar de risico’s. Klinkt toch wel weer als het aloude risicogebaseerd testen.



Naast het nadenken over wat er op welke laag getest moet worden, moeten deze testen ook nog op het juiste moment gemaakt en uitgevoerd worden. Wanneer je te vroeg bent met je testen, kan de hele applicatie nog veranderen, maar wanneer je wat later bent, kan de applicatie al in productie staan. Continuous Integration en Continuous Delivery kunnen alleen goed plaatsvinden als er voldoende dekking is in de geautomatiseerde testset die in de pipeline meedraait. Wanneer deze pipelines goed ingericht zijn, zie je hier de verschillende types testen terug (denk aan: smoketests, api-test, intgratietests, systeemtests,…).

3. Testen

Wat verandert er dan in het daadwerkelijke testen? Tja, dat valt eigenlijk wel mee. We zoeken altijd naar een goede mix van geplande en exploratory tests. Sommige onderdelen moeten wel gepland en bewust gebeuren, terwijl we andere zaken alleen tegenkomen bij daadwerkelijk gebruik van de nieuwe applicatie. Het is net als een stedentrip, overnachtingen wil je bijvoorbeeld wel plannen, maar de precieze invulling van je dagen zie je wel als je de stad aan het verkennen bent.

Een andere belangrijke keuze is die tussen geautomatiseerd en manueel testen. Eerder had ik de pipelines al genoemd en daar hebben we natuurlijk geautomatiseerde tests voor nodig. Maar wees je wel bewust dat automatisering uiteindelijk alleen checks uitvoert en dus nooit dingen rondom het testscenario ziet die een gewone gebruiker wel zouden opvallen. Wanneer de keuze gemaakt is om bepaalde geautomatiseerde tests te maken, moeten deze natuurlijk wel stabiel zijn. Ontwikkelteams raken snel gefrustreerd door false-negatives, en

als tester zit je natuurlijk ook niet te wachten op false-positives.

De meeste recente toevoeging aan het testvak die ik wil aanhalen, is operations. Omgevingen (dus ook productie) moeten door de DevOps teams in de gaten gehouden worden. Ook de tester zal dus met de monitoring om moeten kunnen gaan en de juiste acties ondernemen als er problemen optreden. Daarnaast moet er ook gereageerd worden op vragen van gebruikers. Hier kan een tester door zijn functionele kennis van het systeem vaak van veel waarde zjjn.

4. Conclusie

We zien dus wel degelijk impact op de gebieden mens, proces en testen. De wereld blijft veranderen en daarin zullen ook de testers moeten meebewegen. Waar we veel nadruk zien op de technische affiniteit die nodig is (bijvoorbeeld automatiseren en pipelines), moet de businesskant ook niet onderschat worden. Feit is dat je in de veranderde context niet meer alleen met testen bezig bent.

Jeroen Mengerink