Linkedin discussie: Hoe kan je best geld besparen op testen?

Snelle besparingen

In deze tijden moet iedereen besparen. Dit wordt natuurlijk ook verwacht van een testteam: Waar kan je binnen testen besparen? Er zijn zaken die gelijk harde munten opleveren, zoals: *Stop met testen*, *betaal je testteam minder* of *ontsla een tester*. Dit zou wel tot wat demotivatie kunnen leiden in je team, met al zijn gevolgen nadien.

Testen staat niet op zichzelf en als je aan ons testers vraagt waar we op zouden kunnen besparen, dan komen we toch uit op het efficiënter maken van de testprocessen. Omdat testen afhankelijk is van andere disciplines, vereist dit een samenwerking tussen alle leden van een ontwikkelteam.

Kwaliteit kost geld en levert niets op

Er loopt een discussie op Linkedin over dit onderwerp en daar staan een hoop goede ideeën hoe je in een organisatie op korte- en lange termijn geld kan besparen. En nu komt het rijtje: het verhogen van de kwaliteit van documentatie, het verbeteren van ontwikkelmethodieken, betere communicatie tussen teamleden, afstemming met de klant, goede tooling en afgestemde planning met alle disciplines… Hee? Zijn dit niet gewoon open deuren, als je dit zo leest?

Voor de testwereld zijn deze voorbeelden dit open deuren, het zijn onderwerpen die met het verbeteren van kwaliteit van het uiteindelijke product te maken hebben. Maar iets wat leeft in onze IT wereld: *Kwaliteit is duur*. Besteedt je meer aandacht aan de kwaliteit van een product, zal je meer tijd, mensen en middelen nodig hebben.

Kwaliteit levert geld op

Natuurlijk is dat op korte termijn waarheid. Als je als bedrijf langer denkt te bestaan dan een half jaar is het verstandig om naar andere meningen te luisteren en de volgende woorden te koesteren: Duurzaamheid en kwaliteit in producten zorgt voor besparing op langere termijn.

Een voorbeeld geven is makkelijk: Ik ben geen klusser thuis, maar zal af en toe wat moeten doen. Ik heb in mijn leven al 4 boormachines gekocht van twijfelachtige merken in de uitverkoop voor niet meer dan 10-20 Euro. Waarom vier? Nou, de ene viel gelijk al uit elkaar toen het een schroef zag, de ander deed na een jaar niets meer, de ander had accu's die het niet voor elkaar kreeg om een schroef langer dan 2 CM in een spaanplaat te draaien… Uiteindelijk heb ik een merk gekocht, waar ik al jaren "plezier" van heb. Net zo duur als 4 goedkope boormachines, maar deze werkt nog steeds!

De tips & tricks gevonden op Linkedin

In onderstaande overzicht staan genoemde punten van een discussie op de site Linkedin.com. Deze lijst meenemen in je dagelijks werk? Dit zonder dure verbetertraject of het inhuren van een Test Proces Improvement specialist. De punten zijn onderverdeeld in zaken waar je direct mee kan beginnen, op middel-lange termijn en op lange termijn. Verder nog wat punten die onder andere disciplines vallen dan testen. Dit stuk bespaart je misschien al heel wat Euro's (geen dank hoor ☺)

Directe beginnen met besparen op het testbudget

Korte termijn

1. Bekijk of er niet teveel (test)management op het testproject zit, misschien kan een testmanager of coördinator ook wel meerdere projecten aan.
2. Onderzoek welke licenties van (test)tooling niet gebruikt (Shelf ware) worden en stop met betalen van deze dure licenties.
3. Combineer zoveel mogelijk testgevallen, dit scheelt dubbele testen.
4. Voer prétesten uit, een beperkt aantal testgevallen die de meest gevoelige onderdelen van de software raken. Hiermee kan je snel zien waar de problemen zitten en of je door moet gaan met de testuitvoer op de build die je gekregen hebt.
5. Bekijk waar documentatie versimpeld kan worden (testhandboeken, testplan, testrapportages, dashboard overzichten). Simpele checklists worden beter gelezen en onthouden en hebben meer effect dan handleidingen van 64 pagina's.
6. Maak checklists aan voor alle fasen van een testtraject, dus review checklists, testplan checklists, testuitvoer checklists, testomgeving checklists, et cetera. Deze zijn snel te gebruiken, makkelijk te onderhouden en goedkoop op te stellen.
7. Als tester: Ken de inhoud van de ontwerpen, het is belangrijk dat je goed begrijpt wat de bedoeling is van het ontwerp, zodat je dit snel kan verifiëren in het te testen product.
8. Beschrijf altijd reproduceerbare stappen in bevindingenregistraties. Scheelt uitzoekwerk en is bruikbaar voor her- en regressie testen.
9. Als een testafdeling na een release tijd "over" heeft, schrijf dan alvast testgevallen voor een volgende release aan de hand van change requests die er zijn en worden verwerkt in een volgende versie van het product.
10. Testdata kan ruim van te voren gemaakt worden, als dit niet kan, dan ontbreekt er wat in de ontwerpen.
11. Zet de eindgebruiker vooraan in het traject om mee te helpen met testen. Dan worden eerder de acceptatie test problemen gevonden.
12. Communiceer bevindingen gelijk met de eindgebruiker / klant en laat hem beslissen of deze in de software mag blijven of niet.
13. Evalueer elk testtraject met het gehele ontwikkelteam inclusief gebruiker, per iteratie, zodat men hier allen van kan leren.
14. Werk nauw samen met de ontwikkelaars en test terwijl de ontwikkelaar nog aan het programmeren is.
15. Communicatie verbeteren door wekelijks of zelfs dagelijks een korte voortgang overleg te houden (maximaal 15 minuten per dag). Je hoeft niet in een agile ontwikkelteam te werken om dagelijkse afstemming te verkrijgen.
16. Gebruik het moment van "wachten op een build" door deze tijd te benutten om verbeteringen aan te brengen in de testprocessen.
17. Bekijk de mogelijkheden van Exploratory Testing / Rapid testing. Vooral voor senior testers met veel ervaring in het team. Dit kan veel besparen op documentatie schrijven en bijhouden.

Middel-lange termijn

1. Bekijk waar activiteiten uitgevoerd worden in een testteam die niet direct met testen te maken hebben, teveel documentatie omdat het verplicht is, zijn er statistische gegevens die verzamelt worden en nooit gebruikt worden?
2. Gebruik een binnen je organisatie afgesproken gestructureerd testproces en houd je hieraan.

3. Houdt bevindingenbeheer simpel en de communicatielijnen over bevindingen kort. Geen ingewikkelde flows waar je eerst een cursus voor moet volgen.
4. Ga risico gebaseerd testen: Testen waar het nodig is.
5. Neem genoeg tijd om de teststrategie uit te werken, wat gaan we zwaarder testen, wat minder en wat gaan we niet testen.
6. Bekijk waar je onderdelen van het testtraject parallel kan uitvoeren, systeemtest en acceptatie test kan vaak gecombineerd worden.
7. Schrijf documentatie doelgericht, dus alleen waarvoor het écht bedoeld is, geen zaken beschrijven die je niet gaat uitwerken in een product.
8. Houd je testscripts 'schoon'. In de loop van de tijd kunnen deze uitgroeien tot lijvige documenten. Dubbele tests of tests die niet meer van toepassing zijn worden vaak behouden, maar moeten verwijderd worden.
9. Zaken die elke keer opnieuw worden uitgevonden kunnen in een overkoepelende teststrategie document (checklist) geplaatst worden.
10. Hergebruik bestaande testscripts, testplannen, en andere documentatie. Projecten archiveren wel dit soort zaken, maar men is al lang blij als een project is afgelopen. Archiveren en snel door naar het volgende project. In de archieven van een organisatie kan je veel herbruikbaars terugvinden, als is het af en toe wat lastig zoeken.
11. Testplannen, testgevallen en scripts moeten klaar staan voordat je met de uitvoer begint. Kan dit niet, ga dan Session Based / Exploratory testing uitvoeren.
12. Gebruik een goede planning tool, waar je de activiteiten in het testproces helemaal kan opsplitsen.
13. De testgevallen en testscripts moeten gereviewd worden, ook door de ontwikkelaars en ontwerpers en liefst ook de klant/gebruiker. Als het de juiste reviewers zijn met verstand van het domein waar je in test.
14. Gebruik freeware en open source tooltjes, niet alleen de grote pakketten zoals Rational of Quality Centre. AutoIT is een goed product voor een tester om (bijvoorbeeld) 1000 keer een formulier in te vullen met verschillende data.
15. Zorg dat de juiste tester de juiste dingen doet: Iemand met meer verstand van de techniek achter een product kan wat dieper en meer ontwikkel testen uitvoeren en een testomgeving configureren, een tester met meer ervaring met de business kant kan ontwerpen reviewen , een tester met communicatieve vaardigheden kan product risico analyses uitvoeren, et cetera. De juiste medewerker op de juiste plek werkt het meest efficiënt.
16. Besteedt veel aandacht aan de testomgeving en doe dit zo vroeg mogelijk in een testtraject. Deze moet stabiel en representatief voor de productie omgeving zijn.
17. Resources op de juiste tijd en plaats inzetten, in grotere organisaties kan dat door een centrale manager als "uitzendbureau" te laten fungeren, zodat als er een project met een stilte is, de tester weer ergens anders ingezet kan worden.
18. De testmanager moet ook echt uit de testwereld komen, dit motiveert de testers extra als ze met hun inhoudelijke testvragen bij de testmanager kunnen komen. De testmanager moet vak inhoudelijk kunnen coachen en begeleiden.
19. Houdt brainstorm sessies over je project met alle testers, zodat het niveau van begrip over het testen en het te testen systeem op het zelfde niveau blijft.
20. Grijze gebieden, wie test wat? Zoek de gevallen die niet getest worden en bespreek ze.
21. Traceer bevindingen terug naar de "root cause". Dus de bevinding komt uit het Ontwerp, of door een programmeer fout, of door een omgevings probleem, et cetera. Als je dit bijhoudt kan je op de meest voorkomende root causes actie gaan ondernemen

22. Start een werkgroep van de meest gemotiveerde testers om via bijvoorbeeld brainstorm sessies kleine verbeteringen te introduceren.
23. Zorg dat de bevindingendatabase up to date blijft, dat na het afsluiten van een testsoort (ontwikkeltest, systeemtest, ...) de bevindingen afgesloten of op een “postponed” status staan. Onduidelijkheid over bevindingen kost veel tijd.
24. Bouw traceerbaarheid van ontwerp naar testgevallen. Als een ontwerp verandert moet er een (automatische) rapportage gemaakt worden welke testgevallen aangepast moeten worden.

Lange termijn

1. Ga off-shore testen, met een goed testproces en goede documentatie kan je veel geld besparen op de testuitvoering.
2. Houdt goed bij wat stabiele onderdelen van het systeem zijn door de testresultaten goed te monitoren. Hertest geen stabiele onderdelen. Zo kan je de testaanpak steeds beter fine-tunen en overbodige tests buiten beschouwing laten.
3. Web technologie kan ook via het internet getest worden door enthousiaste testers overal in de wereld. Via www.utest.com kan je een applicatie laten testen door veel testers op veel configuraties. Betaal per bug.
4. Pas het ontwerp en testdocumentatie van aan als de software aangepast wordt. Vaak wordt in de loop van de tijd dit vergeten waardoor de documentatie verouderd en onjuist is. Hierdoor worden fouten in de software niet meer ontdekt.
5. Automatiseer regressie testen, pas op de valkuilen van testautomatisering en besteed veel aandacht aan een goede testtool die bij je organisatie past.
6. Maak een knowledge database over het systeem onder test, zodat wat geleerd is ook weer door anderen gebruikt kan worden.
7. Bekijk of je al klaar bent voor Model Based Testing, dit is een grote investering, maar als je de valkuilen goed in kaart brengt en investeert in goede ontwerpen en een tool voor MBT kan dit op langere termijn veel opleveren.
8. Je kan de keuze maken om testscripts zo gedetailleerd uit te schrijven dat "iedereen" met behulp van deze scripts kan testen. Dit betekent dat collega’s met minde verstand van testen ook mee kunnen helpen. Een bijkomend voordeel is dat dit de inwerktijd voor nieuwe testers inkort. En als je dit beheersbaar en onderhoudbaar kan houden en hiermee op een volwassen niveau zit, dan is testautomatisering de volgende stap.
9. Testers moeten allemaal hun testtraining hebben gehad voor hun niveau.
10. Laat het testteam of kwaliteits afdeling vanaf het begin van een project mee denken en input leveren. Al bij het initiëren van het project. Hierdoor wordt de afstemming beter.
11. Automatiseer het opzetten van een testomgeving. (vooral bij virtuele testomgevingen)
12. Zorg ervoor dat de testers ook als tester denken. Zie test niet als een opstap tot ontwikkelaar of een leertuin voor ontwerpers. Een tester die kan groeien binnen zijn eigen vakgebied binnen de organisatie zal actief meedenken over de efficiëntie van het testen en hier actie op ondernemen.
13. Meet de effectiviteit van testen door statistische gegevens bij te houden. Bijvoorbeeld foundvind effectiviteit ten opzichte van fouten in productie. Hiermee kan je de kosten en berekenen van elk probleem in productie en daarmee opbrengst als deze gevonden wordt in een testtraject.
14. Gebruik versiebeheer tooling en maak goede afspraken. Zo worden zaken niet dubbel gedaan.

Ontwikkelaars

1. Zorg dat je ontwikkelprocessen goed beschreven zijn en gebruikt worden. Een kwalitatief goed proces zorgt voor een goed product en minder testinspanning. Zie voor voorbeelden ISO9000/9001 of Six Sigma.
2. Ga iteratief ontwikkelen in korte cycli, zodat bij elke iteratie het gehele proces wordt 'geoefend' en afstemming sneller gaat.
3. Laat de ontwikkelaars unit testen uitvoeren. Alleen goed gekeurde units die de test hebben doorstaan mogen naar een onafhankelijk testteam.
4. Experimenteer met test-driven development, schrijf het ontwerp vanuit een test perspectief (goed en foutpaden), schrijf automatische tests aan de hand hiervan en ga dan pas de code schrijven voor het eindproduct.
5. Zorg dat de acceptatie criteria voor de eindgebruiker/klant ook helder is. Hoe scherper het doel van het testtraject (waar moet het aan voldoen), hoe meer besparing op tests die niet gedaan hoeven te worden.
6. Ontwikkelaars moeten een testtraining hebben gehad.
7. Maak configuratiemanager of versiebeheerder een aparte functie om configuratiemanagement en de afspraken hieromtrent te bewaken. Het is menselijk om buiten tooling om te gaan werken. Niet iedereen houdt van procedures, dus iemand moet daar scherp op blijven waken.

Planning

1. Een goede planning met een vaste regelmaat zorgt voor rust bij de teamleden. Als men haast heeft door een te krappe planning (chaotisch) dan zal men eerder fouten maken, dat is menselijk. Verder zal de kwaliteitscontrole (reviews en testen) minder opleveren als er build over build wordt “geserveerd” aan het testteam.
2. Laat testers meedenken met budgettaire zaken bij de planning van een project, hierdoor wordt het testbudget bij het starten van het project realistischer.
3. Houdt de planning van de tests goed in de gaten, bij afwijkingen gelijk actie en terugkoppeling naar de projectmanager. (Exception reports, zie Prince2)
4. Laat de testteams een business case maken voor hun testtraject. Kosten en opbrengsten in kaart krijgen is lastig, maar dit geeft wel een goed beeld van wat men kan verwachten.

Documentatie

1. Ontwikkelaars moeten code reviews uitvoeren, het liefst op een gestructureerde manier.
2. Zorg voor goede documentatie, eenduidig en helder voor iedereen (klant, gebruiker, tester, ontwikkelaar) zodat afstemming hierdoor beter wordt.
3. Beschrijf standaarden hoe een ontwerp beschreven zou moeten zijn, voor test is de testbaarheid van belang. Verwerk richtlijnen voor testbaarheid in de standaarden voor een ontwerp.