Het invoeren van testautomatisering

Inleiding

Testautomatisering is een actueel onderwerp binnen de hedendaagse softwareontwikkeling. Vanuit de toenemende vraag naar snelle, kort cyclische product implementaties en de continue vraag naar productverbetering vanuit de eindgebruikers, komt er steeds meer druk op IT-teams om sneller nieuwe functionaliteit op te leveren. Methodieken als Agile Scrum, Kanban, CI/CD en DevOps ondersteunen teams bij het voldoen aan deze wens om te versnellen. De gewenste versnelling heeft impact op alle aspecten van het ontwikkelproces, dus ook op testen. Het effect is dat testen steeds dichter op en in samenwerking met ontwikkeling moet worden uitgevoerd en dat de hoeveelheid handmatig testwerk moet worden geminimaliseerd door invoering van testautomatisering.

Invoeren van testautomatisering klinkt eenvoudig, maar dat is het niet. Testautomatisering is niet het simpelweg automatiseren van de handmatige tests. Ook heerst er het misverstand dat handmatig testen niet meer nodig is en dat het testen goedkoper wordt door testautomatisering. Daarnaast is er niet één standaard manier voor testautomatisering, het is geen silver bullet!

Testautomatisering is complex en kan op vele manieren misgaan. Maar, doe je het goed, dan levert het veel voordelen op.

De voordelen van goed geïmplementeerde testautomatisering

- Snel een volledig inzicht in de kwaliteit van het product bij elke oplevering
- Verhoogde testdekking
- Testautomatisering verhoogt de agiliteit van teams
- Gemotiveerde medewerkers

Snel een volledig inzicht in de kwaliteit van het product bij elke oplevering

Bij elke aanpassing of uitbreiding van het product geven de geautomatiseerde tests snel terugkoppeling of het product goed werkt en of er niet onbedoeld bestaande functionaliteit kapot is gegaan. Hierdoor is er meer grip op - en inzicht in kwaliteit.

Bij een groot aantal handmatige regressietests moet je vaak keuzes maken omdat de tijd ontbreekt om alle tests uit te voeren. Voor geautomatiseerde tests is dit minder relevant. Je bent altijd in staat om de volledige regressietest uit te voeren en hebt dus altijd een volledige regressietestdekking van je product. Mits je uiteraard onbeperkt kunt beschikken over de benodigde apparatuur. De vraag welke tests in de regressietest moeten komen blijft bij testautomatisering wel net zo relevant als bij handmatig testen.

Geautomatiseerde tests verhogen de reproduceerbaarheid en vergemakkelijken het snel opleveren van inzichtelijke testrapportage met gestandaardiseerde metrieken ten behoeve van de besluitvorming.

Verhoogde testdekking

Testautomatisering maakt het uitvoeren van lastige tests (foutafhandeling, edge cases) eenvoudiger, omdat het je in staat stelt het product onder de motorkap te testen. Dit geeft veel meer controle over de testsituatie, omdat je onder de motorkap vaak eenvoudiger de lastige situaties kunt simuleren. Daarnaast maakt testautomatisering het mogelijk om zaken als performance te testen, iets wat erg lastig is om met de hand te doen.

Testautomatisering verhoogt de agiliteit van teams

Geautomatiseerde tests geven snel resultaat, waardoor het snel duidelijk is of een nieuw stuk code naar productie kan. Bij een fout in de code ondersteunt de logging uit testautomatisering bij het debuggen. Last but not least heeft goed gebouwde testautomatisering korte terugkoppellussen, dit wil zeggen dat het uitvoeren van de tests snel verloopt en dat de scope van de test klein is, zodat een fout snel wordt gedetecteerd en gelokaliseerd.

Gemotiveerde medewerkers
Het keer op keer uitvoeren van dezelfde handmatige regressietests is saai en demotiverend werk. Testautomatisering maakt deze werkzaamheden overbodig, waardoor teamleden zich volledig kunnen concentreren op het ontwikkelen en testen van nieuwe functionaliteit die waarde toevoegt aan het product.

Randvoorwaarden voor succesvolle testautomatisering
Bij het invoeren van testautomatisering zijn de volgende zaken van belang:

Testomgevingen en testdata
Vaak vraagt testautomatisering om extra testomgevingen, liefst stabiel met een passende testdataset en volledig onder eigen controle van het team. Bij onvoldoende stabiele testomgevingen zullen de geautomatiseerde tests falen. Niet omdat er een fout in de code zit, maar omdat veranderingen of problemen in de testomgeving of in de testdata de test doen falen. Dit ondermijnt het vertrouwen in de geautomatiseerde test, waardoor ze al snel niet meer gebruikt zullen worden.

Een loosly coupled IT-architectuur
Hoe meer het IT-landschap bestaat uit zwak gekoppelde applicaties en/of services die via goed gedefinieerde en gemanagede interfaces en API's met elkaar communiceren, hoe meer je in staat bent om onderdelen los van elkaar te ontwikkelen en geautomatiseerd te testen. Het geautomatiseerd testen van geïsoleerde onderdelen vermindert de noodzaak van het uitvoeren van veel grote, complexe tests op ketens die per definitie lastig zijn te automatiseren.

Testautomatisering in de taal "van het team"
Kies voor het bouwen van testautomatiseringsframeworks zo veel mogelijk voor programmeertalen die bekend zijn bij de ontwikkelaars. Daarmee zorg je ervoor dat testautomatisering door het hele team gedaan kan worden en het niet een losstaande activiteit van de tester wordt. Testautomatisering in de taal van het team bevordert T-shaping van de teamleden doordat elk teamlid de gebouwde testautomatisering kan onderhouden en uitbreiden. Dit faciliteert dat het hele team zich verantwoordelijk kan voelen voor kwaliteit.

Begin met een pilot
Begin klein met een pilot waarin de tooling wordt geselecteerd en ervaring wordt opgedaan met het embedden van de testautomatiseringsactiviteiten binnen het normale werk van het team. Meet het succes: levert testautomatisering inderdaad het verwachtte resultaat op of zijn er problemen? Los dan eerst de problemen op voordat je testautomatiering gaat opschalen.

Creëer inzicht in de testdekking
Weten dat "alle tests geslaagd zijn" zegt niets over de kwaliteit van het product als je niet weet welke testdekking alle tests gezamenlijk vertegenwoordigen. Vaak zie je dat er bij testautomatisering wordt gestuurd op KPI's als "80% code coverage voor tests". Dit getal zegt helemaal niets over de testdekking van de tests. Code coverage zegt alleen welk percentage van de code geraakt wordt met het uitvoeren van de tests, het zegt dus niet meer dan dat de code uitvoerbaar is, dat hij niet crasht. Kwaliteit ontstaat pas in de tests als er goed wordt nagedacht over welke zaken in de test expliciet moeten worden gecontroleerd tegen een verwacht resultaat. Pas daarom op met het interpreteren van deze KPI want het is heel eenvoudig om eraan te voldoen zonder dat er veel waarde zit in de gebouwde tests. Waar deze KPI wel goed voor is, is het aanwijzen van gebieden met weinig of geen testdekking. Hier moet in detail naar gekeken worden om te bepalen of dit risico acceptabel is of niet.

Daarom is het belangrijk om de testdekking van testautomatisering inzichtelijk te maken door elke test te koppelen aan een herkenbare entiteit van het product. Hoe beter je in staat bent om een product breakdown te maken van je product over alle lagen van je product waarop je met testautomatisering testdekking borgt, hoe beter inzicht je krijgt in de testdekking. Daarmee wordt de waarde van het resultaat van je testautomatisering groter.

In het vervolg van dit artikel gaan we dieper in op de genoemde onderwerpen.

Het is geen silver bullet

Als testen handmatig niet goed gaat, zal testautomatisering dat probleem niet oplossen. Eerst zul je handmatig het testproces op orde moeten maken. Zaken als risico gebaseerd testen en het vaststellen van een teststrategie met een daarbij passende testaanpak moeten geregeld zijn. Pas dan kan testautomatisering daarin geplaatst worden met een scherp oog op de business case, want niet alles is (effectief) te automatiseren. Daarom is testautomatisering ook geen pure IT-aangelegenheid. Gebruik de input van de business en andere belanghebbenden (beheer, architectuur, etc.) bij het vaststellen van de risico's, dan kun je een juiste prioritering vaststellen. Testautomatisering moet doelgericht zijn en niet toolgericht, het is totaal iets anders dan het met een tool geautomatiseerd uitvoeren van de handmatige tests.

Verder is de context waarin je testautomatisering wilt gebruiken erg belangrijk. Het maakt een groot verschil of je het wilt inzetten bij eigen ontwikkelde software of dat je gebruik maakt van een standaardpakket dat geïntegreerd wordt in je eigen landschap en ketens.

Op beide onderwerpen gaan we later nog verder in.

Handmatig testen blijft belangrijk

Het is een misverstand dat testautomatisering alle handmatig tests overbodig maakt. Om een test te kunnen automatiseren moet je precies weten hoe het product zich gedraagt. Een computer kan namelijk alleen de uitkomst van de test vergelijken met het verwachte resultaat dat is voorgeprogrammeerd. Hedendaagse producten zijn echter al gauw zo complex dat het gedrag onmogelijk volledig gespecificeerd kan worden. Voor een deel van het gedrag van het product geldt daarom ook dat je "niet weet wat je niet weet". Foutief en onvoorzien gedrag valt in deze categorie, want de specificatie richt zich meestal op het gewenste gedrag.

Voor productgedrag waarvan je "niet weet dat je het niet weet" geldt dat je het alleen kunt testen door het product op haar gedrag te onderzoeken. Dit vraag om een creatieve, kritische geest die denkt vanuit het gebruik van het product. Deze vorm van testen (exploratory testen) kan alleen uitgevoerd worden door een mens en is niet te automatiseren. Exploratory testen valt onder het kennisgebied Context Driven Testen (CDT). Zie voor meer informatie over dit onderwerp onze website https://www.polteq.com/kennisdeling/goed-exploratory-testen/.

Een good practice voor API/Interface en GUI gebaseerd testen is om eerst het product exploratory te testen. Vanuit de productkennis opgedaan tijdens deze testfase ontstaat dan een goed beeld van waardevolle tests om te automatiseren en toe te voegen aan de geautomatiseerde regressietest.

Het invoeren van geautomatiseerd regressietesten maakt dat de menselijke tester niet wordt belast met het keer op keer uitvoeren van handmatige regressietests, maar dat hij zijn tijd kan besteden aan meer waardevolle exploratory tests.

De juiste aanpak is context afhankelijk

Het maakt voor de strategie van testautomatisering een groot verschil of er sprake is van een ingekocht pakket (al dan niet met maatwerk) of dat de code in eigen huis wordt ontwikkeld.

Bij een ingekocht product verwacht je dat de leverancier de kwaliteit van zijn product geborgd heeft, een beperkte acceptatietest zou moeten volstaan. Maar vaak is er sprake van maatwerk en/of productinrichting die specifiek is voor de klant en die lastig is te testen door de leverancier. Het testen van het maatwerk en/of de productinrichting wordt dan al snel de verantwoordelijkheid van de klant. Omdat je als klant het pakket als oplossing afneemt, kun je het alleen aan de buitenzijde testen. In dit soort gevallen moet testautomatisering op de gebruikersinterface en/of de API's en interfaces worden uitgevoerd.

Bij eigen softwareontwikkeling ben je in staat om testbaarheid van de code en het product in te bouwen en kan testautomatisering op een veel lager niveau in (of op kleinere delen van) de software starten. Testautomatisering wordt op deze manier een all-team effort startend bij de ontwikkelaars. Dit is een belangrijke "good practice" van testautomatisering. Hoe kleiner de codeunits zijn die je geautomatiseerd test, hoe effectiever en efficiënter de testautomatisering is. Dit komt omdat je op deze manier de terugkoppellussen tussen bouwen en testen kort maakt. Elke gebouwde codeunit krijgt zijn eigen stukje geautomatiseerde kwaliteitscontrole in de vorm van unittests. Omdat

de codeunit klein is, is de test ook klein, dus geeft hij snel resultaat. Als de test faalt, weet je precies in welke codeunit de fout zit. Dit verhoogt in sterke mate de efficiëntie van debuggen. Een gedegen testautomatisering op codeunitniveau geeft je de zekerheid dat de codeunits goed gebouwd zijn en doen wat ze doen moeten.

Alleen unittests zijn onvoldoende om garantie te bieden op een goed werkend eindproduct. Het eindproduct is veel meer dan de som van de losse codeunits. De codeunits werken samen en vormen grotere units die specifieke functies in het product leveren. Meerdere functies samen maken de code uiteindelijk tot een product dat waarde heeft voor haar gebruikers. Daarom is het belangrijk om bovenop de fundering van geautomatiseerde unittests een slim gekozen laag van unitintegratietests te leggen, gevolgd door een laag met functionele tests en, als finishing touch, een slim gekozen laag van GUI/ketengebaseerde tests die de waarde van het product voor de eindgebruikers valideert. Al deze extra lagen in je testautomatiseringstrategie zijn belangrijk om inzicht te krijgen in de kwaliteit van je product.

Door te sturen op een zo groot mogelijke testdekking in de lage (onderste) lagen zorg je ervoor dat de benodigde hoeveelheid testautomatisering op de hogere lagen wordt beperkt. Aangezien testautomatisering op hogere lagen van nature trager en meer foutgevoelig is, heeft deze aanpak sterke voordelen.

Testautomatisering is NIET gratis

Testautomatisering is code ontwikkeling. Vaak start het klein en lokaal bij een persoon of een team. Dat is geen probleem voor een pilot om te leren, maar omdat testautomatisering een langetermijninvestering is, zijn onderhoudbaarheid en schaalbaarheid belangrijke aspecten om al vroeg mee te nemen in het ontwerp om de beheeractiviteiten te minimaliseren. Dit vraagt om een goede architectuur en het implementeren van coding standards voor je testautomatiseringscode.

Om testautomatisering goed te kunnen inzetten is testbaarheid van de code een belangrijk aspect. Legacy en bestaande code is vaak niet ontwikkeld met een scherp oog voor testbaarheid, dus om testautomatisering goed te kunnen implementeren is vaak een refactor slag nodig op de code.

Voor het implementeren van testautomatisering zijn tools nodig. De teams moeten zich de tools eigen maken. Daarvoor is specifieke kennis en kunde vereist die niet altijd aanwezig is.

We hebben in de randvoorwaarden voor succes al gezien dat het goed is om met een pilot te beginnen. Voor de pilot zal tijd moeten worden ingeruimd in de sprint. Het is verstandig de pilot goed te refinen:

- Wat voor waarde zal de pilot toevoegen?
- Op welk onderdeel van testautomatisering gaat de pilot zich richten?
- Welke tests moeten geautomatiseerd worden om aan te tonen dat de pilot succesvol is?
- Wanneer is de pilot geslaagd? (acceptatiecriteria)
- Wat is de omvang van het werk (story points schatting)

Door te refinen blijkt vaak dat het verstandig is de pilot op te breken in meerdere user stories. De pilot inspanning moet worden verricht uit de bestaande team velocity dus ander werk zal naar achteren geprioriteerd moeten worden om de pilot mogelijk te maken.

Verder zijn er mogelijk licentiekosten verbonden aan de geselecteerde tools. Zelfs wanneer je kiest voor freeware tools zijn hier kosten mee gemoeid. Beheer van de tools is namelijk vereist, wat ook weer tijd/geld kost

Voor regieorganisaties zonder eigen code-ontwikkeling is het vaak lastig om zelf de benodigde testframeworks te bouwen. We hebben al gezien dat het belangrijk is om kwaliteitseisen te stellen aan testautomatisering. De vraag is nu of er voldoende kennis en kunde is bij de klant van code-ontwikkeling om deze frameworks te bouwen. Als deze kennis niet aanwezig is, moet externe expertise ingehuurd worden.

Last but not least stelt testautomatisering eisen aan de testomgevingen. Gemeenschappelijke testomgevingen zijn per definitie geen goede plek om testautomatisering te implementeren. Vaak zie je dus dat bij het invoeren van testautomatisering er een vraag ontstaat naar meer testomgevingen en virtualisatie van de externe afhankelijkheden. Hieraan is een kostenplaatje verbonden.

Wat krijg je als je het fout doet?

We weten nu dat er best veel komt kijken bij het invoeren van testautomatisering. Dit geeft al aan dat er veel mis kan gaan. Een paar veel voorkomende “bad practices” zijn:

- De bestaande handmatige regressietests worden 1:1 geautomatiseerd
- Alleen de tester is met testautomatisering bezig
- Testautomatisering brengt niet de verwachte tijdwinst

De bestaande handmatige regressietests worden 1:1 geautomatiseerd

Dit is een veelgemaakte fout. Als je te weinig tijd of menskracht hebt om de handmatige regressietest keer op keer uit te voeren lijkt het een logische stap om deze tests te automatiseren. Het idee is niet fout, maar handmatige tests worden veelal op de gebruikersinterface (GUI) uitgevoerd. In de sectie “De juiste aanpak is context afhankelijk” hebben we gezien dat de hoeveelheid testautomatisering op de GUI moet worden beperkt door de gewenste geautomatiseerde testdekking op andere (lagere) lagen in het product te creëren.

Alleen de tester is met testautomatisering bezig

We weten dat testautomatisering zo veel als mogelijk een team effort moet zijn. Als testen het alleen-domein is van de tester is er mogelijk iets mis met het eigenaarschap voor kwaliteit binnen het team. Of er is gekozen voor een programmeertaal voor testautomatisering die alleen de tester kent. Last but not least, mogelijk is het team onvoldoende op de hoogte van de mogelijkheden en voordelen van vroegtijdige, kortcyclische testautomatisering.

Testautomatisering brengt niet de verwachte tijdwinst

Er is veel onderhoud nodig om de tests draaiend te houden. Als de tests draaien, draaien ze wel sneller dan als de tests handmatig zouden worden uitgevoerd, maar door het vele onderhoud dat de tests keer op keer behoeven, gaat deze tijdswinst gedeeltelijk of zelfs volledig verloren.

Dit resulteert in ongemotiveerde teamleden, omdat ze keer op keer de kapotte tests moeten repareren. Al snel ontstaat dan de neiging om de kapotte tests maar “even uit te zetten”. Met het uitzetten van een test zeg je eigenlijk “ach, deze test is niet zo belangrijk”. Dit is zorgelijk, want dit zegt eigenlijk dat de waarde van de gemaakte test niet heel hoog is!

Daarnaast geldt bij het even uitzetten van een test de regel “als één schaap over de dam is, volgen er meer”. Dit wil zeggen dat als het besluit is genomen om een test uit te zetten, het daarna makkelijker is om nog een test uit te zetten. Voordat je het weet zijn steeds meer tests “even uitgezet” en wordt de benodigde inspanning om alle uitgezette tests te repareren groot. Er ontstaat op deze wijze al snel een test debt die lastig is om in te lopen.

Polteq kan u helpen

Zoals we gezien hebben is het inrichten van testautomatisering een ingewikkeld vraagstuk, waarbij met veel aspecten rekening moet worden gehouden om succesvol te zijn. Herkent u één of meer van de genoemde foutsituaties in uw eigen organisatie, dan is dat een indicatie dat de aanwezige testautomatisering suboptimaal is. Het is dan zeker goed om de bestaande testautomatisering eens tegen het licht te houden.

Indien gewenst kan Polteq u hierbij op allerlei vlakken ondersteunen:

- Opleidingen
- Consultancy en testverbetering
- Inrichten van testautomatisering

U vindt ons hier: www.polteq.com/