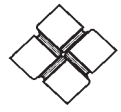
De karakteristieken van een modern testproces

20 praktijkcases over testen in de nieuwe wereld

Auteur
Jan Jaap Cannegieter

Co-auteurs
Derk-Jan de Grood
Edwin van Loon
Gerben van Oene
Han Niessing
Jeroen Rosink
Jeroen van Deelen
Lucas Sikking
Mark Franssen
Martijn van Werven
Michel van 't Zant
Miranda Kerpel
René van Veldhuijzen
Ruud van Berkum
Steven van Voorst
Tijs Latiers
Maike Jansen-Oosterbaan



Eburon
Utrecht 2020

Fotografie: Erik van der Burgt, VRBLD photofilm
Omslagontwerp: Textcetera, Den Haag
Grafisch ontwerp: Textcetera, Den Haag
Productie en uitgave: Uitgeverij Eburon, Utrecht, www.eburon.nl

ISBN 978-94-6301-288-1

Inhoud

Deel **een**

Deel **twee**

Inleiding

In 1996 deed ik mijn eerste testproject. Bij de Belastingdienst in Apeldoorn. Toen ik daar startte zei een ontwikkelaar, destijds bouwer genoemd, tegen me: 'Ben jij de nieuwe tester? Wat vervelend voor je dat je tester bent, wat doe je over vijf jaar?' Hij was ervan overtuigd dat vijf jaar later, in 2001 dus, er geen testers meer nodig waren. Als reden gaf hij de opkomst van Rapid Application Development en 4GL taken. Sindsdien is het einde van testen voorspeld als gevolg van beter opgeleide ontwikkelaars, development tooling, Agile, testautomatisering, low code platforms en nu kunstmatige intelligentie. En waarschijnlijk vergeet ik dan nog wel een paar ontwikkelingen van de laatste jaren.

Sinds 1996 is het aantal testers echter aanzienlijk gestegen. Er zijn talloze boeken geschreven, meer professionals zijn gecertificeerd, er zijn heel veel testconferenties bij gekomen en het vakgebied floreert als nooit tevoren. Testing is here to stay! Keer op keer blijkt maar weer dat testen iets toevoegt aan organisaties in het algemeen en systeemontwikkeling in het bijzonder. En volgens mij is er een meer fundamentele reden dat testers waarde toevoegen en dat is dat ze anders tegen de systemen die worden ontwikkeld aankijken en dat je die vaardigheid nodig hebt in systeemontwikkeling om een systeem van voldoende kwaliteit te maken.

Maar testen is sinds 1996 wel heel erg veranderd! We zitten nu (vaak) niet meer in een apart testteam maar zijn onderdeel van het ontwikkelteam, we gebruiken allerlei soorten tooling om efficiënter en effectiever te werken en we hebben niet meer een aantal weken aan het eind van een ontwikkeltraject de tijd om te testen, om maar wat veranderingen te noemen. Maar in veel organisaties zijn de methoden die we gebruiken vaak nog in grote lijnen dezelfde als de methoden

van eind vorige eeuw. De traditionele methoden zijn ontwikkeld in een tijd waarin zaken zoals fasering, documentatie, voorbereiding en procesmodellen belangrijk waren. Nu zijn zaken zoals flexibiliteit, businesswaarde, pragmatisme en snelheid veel belangrijker. Aan de buitenkant lijken deze methoden zich wel ontwikkeld te hebben maar vaak zijn de onderliggende uitgangspunten, de karakteristieken van deze methoden, nog hetzelfde. We moeten dus op een andere manier testen. Met een aantal zeer gewaardeerde collega's van Squerist heb ik hierover een aantal avonden zitten discussiëren en hebben we ideeën uitgewisseld. We kwamen tot de conclusie dat er geen enkele methode is die altijd, in iedere situatie goed toepasbaar is. En die methode kán ook niet gemaakt worden. Waarom? Omdat iedere organisatie anders is, ieder project of team uniek is en ieder systeem anders is. Kortom: omdat iedere situatie anders is en we dus het testen altijd anders aan moet pakken. Is er dan helemaal niets te zeggen over modern testen? Ja, zeker wel! Er is een aantal karakteristieken of kenmerken te onderkennen die we veelal terugzien bij modern testen. Volgens ons zijn die karakteristieken doelgerichtheid, flexibiliteit en snelheid. Een testproces dat daaraan voldoet, is ook écht anders dan een traditioneel testproces wat meer gericht was op voorbereiding, documentatie en gedefinieerde processen. Een modern testproces is echter niet te beschrijven in een altijd toepasbare en alles omvattende methode, procesmodel of iets dergelijks. Er is echter wel bij testers, business managers, IT-managers en ontwikkelaars behoefte om inzicht te hebben in hoe modern testen eruit kan zien. Hoe maak je dan duidelijk hoe een modern testproces er in de praktijk uitziet? Door de praktijk te beschrijven! Door te laten zien hoe deze karakteristieken in de praktijk zijn toegepast. En dat is precies wat je aantreft in dit boek, een groot aantal praktijkcases waarin minimaal één medewerker of opdrachtgever van Squerist participeerde en waar een aspect van modern testen is toegepast. Omdat we bij Squerist niet alleen praten over modern testen maar dat ook toepassen. Alle cases zijn in de vorm van artikelen eerder gepubliceerd in een vakblad of op een website. Dit maakt de cases los leesbaar. Als lezer kun je die artikelen lezen die voor jou interessant zijn. Je kunt dit boek natuurlijk ook 'van kaft tot kaft' lezen. Voordat we aan de cases beginnen vind je eerst een beschrijving van de karakteristieken van modern testen; een uitwerking van doelgericht, flexibel en snel. Dit gedeelte biedt een kader voor modern testen en voor de praktijkcases. Het is wel wat theoretischer en je kunt het gerust overslaan als je meer geïnteresseerd bent in de praktijk.

Over de doelgroep

Een belangrijke vraag bij ieder boek is voor wie je het boek schrijft, welk kennisniveau je verwacht van de lezers. Om dit helder te maken eerst even voor wie dit boek niet bedoeld is. Ten eerste is dit boek niet bedoeld voor doorgewinterde testprofessionals die zeer diepgaande kennis van testen hebben en op de hoogte zijn van alle moderne ontwikkelingen. In dit boek staan, met uitzondering van deel 1 en twee artikelen aan het eind, praktijkcases beschreven. Dingen die nu al gebeuren. In dit boek vind je dus geen abstracte toekomstvisie. Wat je wel vindt, zijn praktisch bruikbare zaken die zich bewezen hebben in 'het veld'. Dit boek is ook niet geschreven voor mensen met geen of weinig kennis van testen. Algemeen aanvaarde testconcepten en -termen worden als bekend verondersteld en niet verder uitgelegd.

Dit boek is bedoeld voor testers die nog niet in een moderne omgeving werken en voor testers die wel in een moderne omgeving werken maar nog niet helemaal weten hoe ze dit het beste kunnen doen. Daarnaast is het voor managers en medewerkers geschreven die wel veel met testen te maken hebben maar niet dagelijks testen. Denk hierbij aan business management, product owners, scrum masters, scrum coaches, IT managers, project managers en delivery managers. Zodat zij beter inzicht krijgen in wat ze wél en wat ze niet van testen in een moderne omgeving kunnen verwachten.

De missie van Squerist is 'Verder brengen'. Verder brengen van onze opdrachtgevers, onze medewerkers en ons vakgebied. Dit boek past hierin. Wij van Squerist willen jou als opdrachtgever, medewerker of anderszins testprofessional verder brengen. En om dat te bereiken, delen we onze visie, kennis en ervaring ook graag met jou.

Veenendaal, januari 2020.
Jan Jaap Cannegieter

Deel een

Karakteristieken van een modern testproces

In de inleiding heb je kunnen lezen dat de karakteristieken van modern testen volgens Squerist doelgericht, flexibel en snel zijn. In dit eerste deel wordt uitgelegd wat we daaronder verstaan. Daarmee is dit het enige theoretische deel van dit boek.

Doelgericht

Testen kost geld, veel geld. De schattingen lopen uiteen van 20% tot 50% van de systeemontwikkelkosten. Daar moeten dus opbrengsten tegenover staan. Of anders gezegd: testen moet een doel dienen. Ongeacht of je in een traditioneel watervalproject of in een Agile organisatie werkt: het doel, de verwachte toegevoegde waarde en verwachte opbrengst van testen moeten helder zijn.

Nu is het doel van systeemontwikkeling in iedere situatie anders. Het doel van het testen is afgeleid van het doel van het systeem. Testers moeten zich laten leiden door deze systeemdoelen. Met die tip zou het kenmerk doelgericht klaar kunnen zijn; het doel van testen is afgeleid van het systeemdoel. Er is echter wel meer te zeggen over doelgerichtheid in testen. Op een hoog abstractieniveau kun je namelijk een aantal zaken aantonen met testen; we kunnen vaststellen of een systeem aan een set eisen voldoet, we kunnen de risico's van implementatie inzichtelijk maken en we kunnen inzichtelijk maken of de beoogde doelstellingen zijn gehaald. Op basis van deze driedeling onderkennen we de volgende vormen van testen:

- Checken
- Risicogebaseerd testen
- Waardegebaseerd testen

Checken

In het kader van testen wordt checken soms gedefinieerd als het proces van het uitvoeren van evaluaties door algoritmische beslissingsregels toe te passen op

specifieke waarnemingen van een product[1]. Door middel van checken stel je dus vast of een systeem of een wijziging voldoet aan een set gedefinieerde eisen; er is sprake van een heldere verwachte uitkomst. Voorbeelden van die set gedefinieerde eisen zijn een set requirements, branche-specifieke eisen, contractuele afspraken en wetgeving. Checken ligt dichtbij verifiëren: is het systeem juist gebouwd, waarbij 'juist' bepaald wordt door een set gedefinieerde eisen.

Om het nog wat concreter te maken, volgt hier een voorbeeld. Bij diverse (overheids-)organisaties draaien zogenaamde maatschappijkritische systemen. Als die uitvallen, zijn de gevolgen voor de maatschappij groot. Denk aan verkeer wat (echt) vast staat, vliegtuigen die niet meer op kunnen stijgen, elektriciteit die uitvalt of communicatiemiddelen die niet meer functioneren. Om de risico's die daaruit voortvloeien te beperken, heeft de overheid eisen opgesteld waaraan bepaalde systemen moeten voldoen. En als je aan een dergelijk systeem werkt, moet je vaststellen, ofwel checken, of je aan die eisen voldoet. Einde discussie. Deze activiteit kun je zien als testen en wordt bij veel overheidsorganisaties ook aangeduid als testen. Maar feitelijk is dit checken of je aan een gedefinieerde set eisen voldoet.

Ook in commerciële organisaties kan checken belangrijk zijn, ook hier een voorbeeld. Sommige systemen worden door een leverancier gemaakt. Denk aan softwarehuizen, IT-dienstverleners en dergelijke. De opdrachtgever heeft vaak een set requirements waaraan voldaan moet worden. Het moet op een specifiek platform draaien, we moeten instructies hebben voor de gebruikers, als er een transactie plaatsvindt moet dit worden vastgelegd in een log en dergelijke. Een ander voorbeeld zijn compliance-eisen waar in sommige gevallen aan voldaan moet worden. Als je in één van de bovenstaande situaties zit, moet je natuurlijk wel vaststellen of je aan die eisen voldoet: checken dus.

Checken is lang niet altijd risicogebaseerd. Er zijn situaties dat je vast moet stellen of het systeem wel of niet aan een set eisen voldoet. Dan doe je geen risicoanalyse of impactanalyse, maar maak je testscripts en voert die uit. Wel moet je aan het begin van het traject vaststellen of je moet checken en zo ja op basis van welke referentiekaders. Dit kun je een risicoanalyse noemen, maar is eigenlijk meer een inventarisatie. Bij checken of aan een set (inhoudelijke) requirements wordt

1 Zie: https://www.satisfice.com/blog/archives/856

voldaan, kun je nog in enige mate risicogebaseerd werken, maar vaak is dit ook gewoon een set scripts maken en vaststellen of eraan voldaan is of niet. Gewoon recht-toe-recht-aan.

We zeggen niet dat je in ieder testtraject moet checken, soms is er geen wet- of regelgeving, geen set requirements of iets dergelijks waarvan vast moet worden gesteld dat het systeem daaraan voldoet. Wij vinden wel dat testers zich bij iedere situatie moeten afvragen of checken van belang is en zo ja, hoe dit georganiseerd gaat worden.

Risicogebaseerd testen

Bij risicogebaseerd testen test je het systeem, afhankelijk van het risico, diepgaander of minder diepgaand. Stel dat je een kleine, geïsoleerde aanpassing maakt op een weinig gebruikte app. De aanpassing wordt gemaakt door zeer ervaren ontwikkelaars die altijd goed werk hebben geleverd. En als die aanpassing niet werkt merken weinig gebruikers dat. Dan is het prima om de aanpassing niet te testen, de risico's zijn in dit voorbeeld erg laag. Nu een andere situatie: stel dat een bank een wezenlijke aanpassing maakt aan haar online banking app die diep ingrijpt in de app en de achterliggende systemen. En als deze wijziging niet helemaal goed wordt doorgevoerd kan dit leiden tot het niet functioneren van de app of kan er een security lek geïntroduceerd worden. In een dergelijk geval mag ik toch hopen dat dit getest wordt. En goed ook! Hoe groter het risico, hoe belangrijker het is dat het goed getest wordt.

De essentie van risicogebaseerd testen is dat je inzichtelijk maakt wat de risico's van een systeem of aanpassing zijn en hoe je die risico's af gaat dekken. En eigenlijk kunnen wij ons geen situatie voorstellen waarin het niet nuttig is om, naast het inventariseren welke checks plaats moeten vinden, een risicoanalyse te doen. Uitkomst van de analyse zou kunnen zijn dat er geen risico's zijn en dan ga je dus niet testen. Die conclusie is op zichzelf al waardevol. Maar in de praktijk zijn er altijd risico's en die kun je maar beter van te voren kennen en erover nadenken hoe je die risico's afdekt.

Nog wel even een punt van aandacht. Bovenstaand verhaal zou kunnen impliceren dat we uitgaan van grote watervalprojecten. Van tevoren is alles bekend en in te schatten en we kunnen aan het begin een uitgebreide grote risico-inschatting doen. Maar risicoanalyse kan en moet je ook in Agile doen! Continu. Iedere release

en sprint weer. Is dit veel werk? Nee hoor. Wijzigingen in Agile zijn kleiner, dus de inschatting ook. Daarnaast helpt het maken van de risicoanalyse bij het inschatten van de inspanning om de wijziging te bouwen en te testen. Even in vaktermen: je kunt de risicoanalyse onderdeel maken van de refinement. Indien sprake is van een organisatie waarin meerdere Agile teams parallel werken, kun je de risicoanalyse ook onderdeel maken van de releaseplanning.

Risicogebaseerd testen houdt dus in dat je risico's inschat en afhankelijk van de uitkomst maatregelen neemt.

Waardegebaseerd testen

Eerder hebben we de stelling geponeerd dat testen eigenlijk best veel geld kost. Laten we die stelling even oprekken en zeggen dat nieuwe systemen en aanpassingen aan bestaande systemen nu eenmaal veel geld kosten. En dan is het goed om vast te stellen of het geld dat een organisatie in de systeemontwikkeling

steekt ook de verwachten voordelen brengt of heeft gebracht. Dat is waardegebaseerd testen: vaststellen of de te verwachte voordelen ook daadwerkelijk worden gehaald of zijn gehaald. Lukt het om 50% sneller een order in te voeren? Kunnen klanten nu direct onze actuele voorraad zien? Of ingewikkelder: wordt de besparing van 10% gerealiseerd? Waardegebaseerd testen ligt dicht bij valideren: is het juiste systeem gebouwd?

Voor waardegebaseerd testen zijn de systeemdoelen of business case erg belangrijk, niet de specificaties van het systeem. Testen op basis van specificaties is checken, testen op basis van (systeem)doelstellingen waardegebaseerd testen. En waardegebaseerd testen verschilt écht van 'normaal' testen. Het is veel minder het maken en uitvoeren van testgevallen. Het is onderzoeken, verzamelen van informatie en inschatten wat deze informatie ons vertelt. Waardegebaseerd testen gaat dan ook verder dan traditioneel testen. Naast testers zijn daar dan ook vaak andere stakeholders bij betrokken zoals IT-management, business management, de product owner en finance. Agile teams kunnen hierin ook de verantwoordelijkheid nemen. Het is dus veel breder dan traditioneel testen.

De vraag is wanneer je goed waardegebaseerd kan testen. Eigenlijk zou je het pas écht kunnen doen als je een tijdje in productie bent. Maar volgens Squerist kun je van veel dingen ook al een inschatting maken voordat je in productie gaat, bijvoorbeeld als je aan het testen bent. Je hoeft dus niet met waardegebaseerd testen te wachten tot het systeem in productie is. En door waardegebaseerd testen vóór in productiename te doen, kun je ook eerder corrigerende maatregelen nemen als dat nodig is. Laten we voorbeelden geven. Het eerste voorbeeld is een systeemontwikkeltraject waarbij, na de implementatie van het systeem, ongeveer 20% van het personeel overbodig wordt. Als het systeem definitieve vormen begint aan te nemen, kunnen testers die veel inzicht hebben in de werking van het systeem en de bedrijfsprocessen best inschatten of dit gehaald wordt. Testers kunnen onderzoeken hoeveel handelingen vroeger handmatig gebeurden en op basis van het testen kunnen ze vaststellen hoeveel handelingen in de het nieuwe systeem handmatig worden gedaan. Op basis daarvan kan worden vastgesteld of de doelstelling wordt gehaald.

Een tweede voorbeeld is de situatie die een van de auteurs heeft meegemaakt. Het gaat hier om een organisatie die een aantal systemen verving door nieuwe systemen om vervolgens de nieuwe systemen aan elkaar te koppelen om zo meer verkoop-leads te genereren. Tegen het eind van het project rapporteerden

de teams aan de stuurgroep dat de systemen live konden, de risico's waren aanvaardbaar. Niet alle functionaliteit was gerealiseerd, maar het primaire proces werd goed ondersteund door de nieuwe systemen. We hadden gejuich van de business verwacht. Maar die zeiden: we gaan nog niet live. Want het doel was uitwisselen van informatie zodat we verkoop-leads krijgen. En dat was nog niet gebouwd. We konden dus een aantal goed draaiende systemen vervangen maar dat leverde de business geen waarde op. We hadden alleen risicogebaseerd gedacht, niet waardegebaseerd. Er was nog geen waarde gecreëerd.

Waardegebaseerd testen is eigenlijk op een andere manier tegen de uitkomsten van je testen aankijken en dit kan voordat je in productie gaat (verwachten we met deze versie de verwachte voordelen te halen?) en nadat je in productie bent gegaan (halen we daadwerkelijk de voordelen?). Waardegebaseerd testen is denken vanuit de doelstellingen van het systeem. Dat gebeurt nog (te) weinig. Hier liggen kansen op het leveren van meer toegevoegde waarde als testers. Want wees eerlijk: hoeveel testers rapporteren op basis van de systeemdoelen?