FAST DELIVERY ON A SLOW TRAIN

Door Marc van 't Veer ● marc.vantveer@polteq.com

De metafoor

'Hoe past het snelle metrosysteem op het traject van een langzame goederentrein?'. Dat is een metafoor voor het combineren van verschillende ontwikkelmethodieken binnen één project met één releasedatum. Daarbij gaat het niet om een voorkeur voor een bepaalde ontwikkelmethodiek, maar om een beschrijving van een situatie die steeds vaker zal gaan voorkomen. Welke moeilijkheden kom je tegen en welke oplossingen kun je toepassen, zodat je als tester je werk goed kunt doen?

De dienstregeling

De dienstregeling van een goederentrein, van emplacement tot eindstation, met alle tussenstations, wordt lang van tevoren vastgelegd. En als je mee wilt, zul je je daaraan moeten houden. De metro daarentegen neem je naar behoefte, bijvoorbeeld omdat het regent. Hoe stem je dan de dienstregeling van de metro af op die van de goederenlijn tussen Rotterdam en het Duitse achterland? Een metro vertrekt bijvoorbeeld elke tien minuten en een specifieke goederentrein eens per dag. Een metroreis vraagt qua timing veel minder planning. Als niet iedereen in de metro past, dan kunnen ze met de volgende mee, met een maximale wachttijd van die tien minuten. Als echter jouw containers niet met de goederentrein meekunnen, dan heb je een serieus probleem, want dan komen ze minstens een dag te laat aan. Die goederentrein moet dus strak worden gepland.

Maar wat gebeurt er als een machinist de metro mist of in een verkeerde wagon stapt, of een wissel stuk is? Dan kan, net als bij de Nederlandse Spoorwegen op een winterse dag, de hele dienstregeling vastlopen. Ook bij de metro moet je het een en ander plannen.

Figuur 1. Metro op de Betuwelijn?

Het project

Het totale project waar ik het over wil hebben, werd uitgevoerd door verschillende leveranciers. Elke leverancier volgde zijn eigen methodiek, zijn eigen 'spoor'. Met meer dan honderd medewerkers werd een nieuw programma live gezet. De backend-systemen zijn in waterval ontwikkeld, de (mobile) website in iteraties en de mobile app Agile. Elke leverancier was verantwoordelijk voor het uitvoeren van systeemtesten en het leveren van support tijdens ➔

de integratiefase. In totaal bestaat de end-to-end-keten uit meer dan twintig applicaties die onderling communiceren via een enterprise service bus. Het totale programma had één centraal go-live-moment.

De uitdaging

Voorwaarde om binnen het project te kunnen overleven is basiskennis van alle drie ontwikkelmethodieken. Maar het is best een uitdaging om je aandacht te verdelen over niet alleen de lange termijn (de waterval) en de korte termijn (Agile), maar ook nog over de losse iteraties op zich. Je moet aan de ene kant dagelijks succesvol weten te testen binnen het Agile project, maar ook nog genoeg tijd overhouden voor de voorbereiding van de watervaltesten. Naast het verschil in planningshorizon verschillen de methodieken ook in de Definition-of-Done. Wanneer ben je bijvoorbeeld klaar binnen een watervalmethodiek (testcases staat op 'passed'?), binnen een iteratieve aanpak (alle iteraties geïntegreerd getest?) en binnen Agile methodiek (de user story werkt?).

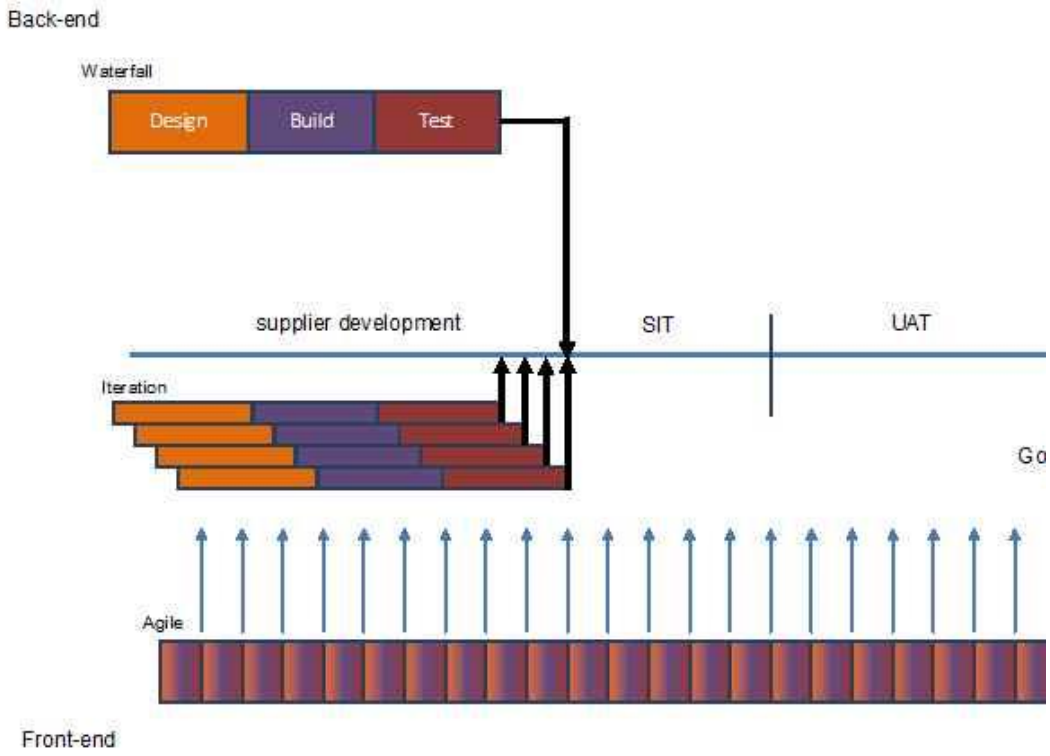
Een tester moet dagelijks schakelen tussen parallel lopende releases en bepalen welke strategie wordt gevolgd om kwaliteit te leveren. Wanneer moet je overstappen, hoeveel overstaptijd of wachttijd is er, of moet je iedere keer een sprintje trekken om soepel te kunnen doorreizen? Het totale dienstrooster en de frequentie van rijden geven het tempo aan, waarin gewerkt moet worden door een tester. Het dienstrooster is de releasekalender. Deze kalender moet kloppen als je hart (heartbeat), zodat kan worden bepaald wanneer je welke trein moet nemen, wanneer je kunt gaan zitten, moet gaan staan of beter kunt blijven staan. Welke taken moet je uitvoeren voor welke release, wanneer moet je voorbereiding klaar zijn, hoe moet je een taak uitvoeren en wie voert een testcase uit op welke omgeving? De uitdaging zit in het flexibel schakelen tussen de verschillende leveranciers, methoden, releases, deadlines, taken en begrippen.

Problemen, risico's en wat het zo complex maakt voor een tester

Een tester is van nature sceptisch, (positief!) kritisch en gezond wantrouwend van karakter. Het eerste wat ik deed, was vragen stellen om de organisatie en het gehele systeem te leren kennen. Voorbeelden: Wat gebeurt er als een defect wordt gevonden, wat als er een verandering in de markt is, een extra change nodig is, een afhankelijkheid tussen systemen is gemist, een deadline wijzigt omdat deze niet wordt gehaald of als er een hele release extra nodig is? Door deze vragen wordt meer duidelijk over hoe de raderen draaien en waar het gaat piepen of kraken.

Problemen

De organisatie waarvoor we dit project uitvoerden, heeft een 'natuurlijke' focus op de logistieke keten en de processen daaromheen. De IT-processen worden bestuurd door middel van een watervalmethodiek. Kijk je als tester naar hun markt, dan zie je veel competitie, stellen de klanten hoge eisen en is 'time to market' veel belangrijker dan het hebben van alle features. Als tester was ik verantwoordelijk voor het coördineren van de gebruikersacceptatietest (UAT), in dit geval zowel voor de business (intern) als voor de eindgebruikers in de markt (extern). De eisen van de eindgebruikers spelen een cruciale rol, wil je als totale organisatie succesvol zijn. Het voelt als tester alsof je een strijd moet leveren om de eisen van de eindgebruiker gerealiseerd te krijgen. Naast de strijd tussen de eigen organisatie en de markt, bestaat er ook een gevecht tussen de verschillende leveranciers. Wiens visie sluit het best aan op de organisatie en de markt, welke oplossing wil iedere leverancier het liefst geïmplementeerd hebben, welke contracten spelen een rol, en welke politieke krachten worden er uitgeoefend? Deze strijd heeft veel invloed op hoe open, soepel en meewerkend alle partijen naar elkaar zijn. ➔



Figuur 2. Parallel ontwikkelen zonder afstemming

Risico's

Niet alleen een tester moet continu zijn aandacht verdelen; dat geldt ook voor een product owner. Een product owner kan niet deelnemen aan alle parallel lopende teams tegelijk. Daarom was in dit geval de product owner niet één persoon, maar een groep die de regie voert over de gestelde eisen. Doordat verschillende onderdelen van het totale systeem door verschillende leveranciers werden ontwikkeld, werden besluiten over de architectuur, interfaces en business rules los van elkaar genomen. Die moeten dan natuurlijk wel onderling worden afgestemd. Het risico is dan levensgroot dat niet alle specificaties van en naar alle leveranciers met elkaar in sync zijn.

Naast de eisen van de business komt de uiteindelijke eindgebruiker in elke methodiek op een ander moment binnen en speelt een andere rol. Binnen de watervalmethodiek is eindgebruiker het eindstation van de ontwikkelcyclus, binnen de andere methodieken wordt de eindgebruiker veel meer en veel vroeger betrokken; hier is de eindgebruiker echt een klankbord. De eisen van de eindgebruiker kunnen de eigen specificaties in een ander daglicht stellen, prioriteiten verschuiven en nieuwe specificaties noodzakelijk maken.

Complexiteit voor een tester

De volgende punten maken het complex(er) voor een tester om in deze situatie te werken:

- Continue wisselen tussen proces en product view;
- Kortetermijn- versus langetermijnfocus;
- Eén wijziging in de planning kan een lawine aan andere wijzigingen veroorzaken;
- Meer machinisten op de trein door een matrixorganisatie;
- Complex communicatieproces;
- Zeer frequente (wekelijkse) software merges;
- Parallel ontwikkelde releases; ➔

- Vaak heel hard rennen voor meerdere teams en dan een dag stil staan;
- Continu scope-discussies en verschuiven van verantwoordelijkheden.

Aan het einde van de dag, week, maand, als alles weer is afgestemd, vraag je je als tester af: is er nog tijd genoeg om alle tests uit te voeren en hoe krijg ik dat gepland? Als tester signaleer je defects en heb je veel invloed op de wijze waarop leveranciers samenwerken. Je bent de boodschapper van het resultaat van veel hard werk. Op welke manier kun je bijdragen aan het soepeler afstemmen van de verschillende manieren van werken?

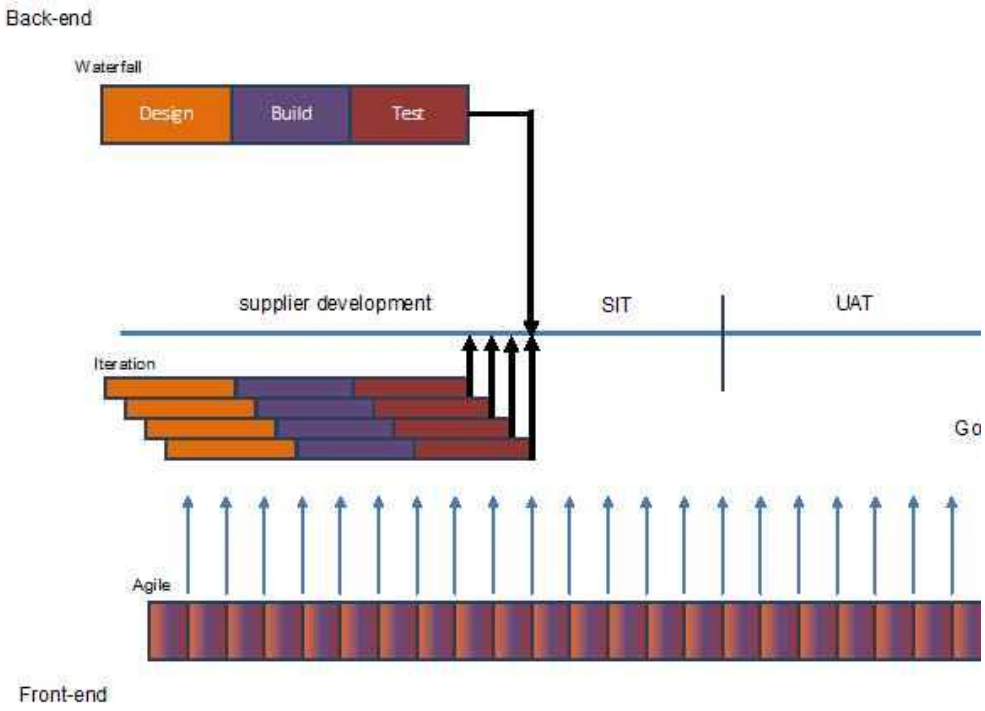
Oplossingen

Als tester wil je samenwerken met en service leveren aan alle leveranciers. Ik geloof dat alleen door samen te werken kwaliteit kan ontstaan. Daarnaast geloof ik er niet in dat één ontwikkelmethode de 'beste' is voor alle leveranciers. Het doel is niet om het perfecte proces te volgen, maar om gezamenlijk een kwaliteitsproduct op te leveren. De oplossingen, die je als tester helpen om te kunnen werken in zo'n complexe situatie, zijn volgens mij te vinden in de flexibiliteit van schakelen in je hoofd, in de methoden die je toepast, in de omgevingen en de tooling die worden gebruikt. Daarnaast kun je bepaalde zaken zo organiseren dat ze de dagelijkse praktijk van een tester helpen (enablers). Ook kunnen aanpassingen in de drie ontwikkelmethoden van pas komen.

Enablers

Voorbeelden van enablers die voor mij belangrijk zijn gebleken, zijn:

- Defects centraal gemanaged (applicatieproces);
- Een team van product owners voor alle partijen en applicaties;
- Meerdere volledig ingerichte en onderhouden testomgevingen;
- Een standaard raamwerk voor alle interfaces voor alle applicaties;
- Een centraal register van alle interfaces;
- Een centrale hartslag in de vorm van een releasekalender;
- Een centraal discussieplatform voor planning, issues, vragen en designkeuzes;
- Een overzicht van afhankelijkheden (planning en applicaties);
- Een geautomatiseerde regressietestset (inclusief de benodigde tools);
- Overzicht van de voortgang binnen de teams;
- Afstemmingsproces tussen alle teams. ➔



Figuur 3. Centrale releasekalender

Afstemming methodieken

Behalve dat het project voor afstemming kan zorgen, kan dit ook plaatsvinden binnen elke ontwikkelmethodiek zodat deze beter aansluit op de rest. Ik hanteerde verschillende opties om waterval, iteratief en Agile methoden samen te voegen:

- Dezelfde tester zit in het Agile-team en doorloopt vervolgens alle volgende fasen in de waterval;
- Testers in koppels, allebei in een Agile- en in een watervalteam, en onderling wordt het werk verdeeld (input en feedback naar beide teams);
- Meerdere Agile-rondes, daarna een integratiefase en overdracht naar waterval;
- Kleine scope in Agile en grotere scope (ketenintegratie) in waterval;
- Vroege betrokkenheid in het watervalproject om voorbereidend werk te doen en ruimte te maken voor deelname aan het Agile team, daarna weer doorgaan in de watervalfasen.

Ook een tester moet zich wapenen voor een strijd binnen het project. Kijk daarbij naar je kennis (gears in je hoofd) en naar zijn je kunde (gears in je methoden).

Gears

Hier is een aantal voorbeelden van de vlakken waarop ik als tester moest schakelen om in alle teams mee te komen:

- Kennis nemen van (alle!) gebruikte termen en begrippen;
- Kunnen omgaan met wijzigingen tussen verschillende teams;
- Kunnen wisselen tussen een proces- en productfocus;
- Verschillende rollen, taken en verantwoordelijkheden hebben;
- Tegelijkertijd een Scrum-master en meerdere testmanagers hebben;
- Kunnen reizen op de releasekalender. ➔

Het belangrijkste is dat ik mijn werk tegelijkertijd zowel traditioneel gefaseerd kan uitvoeren als deelnemend aan een Agile-team. Het is jongleren in taken die gelijktijdig moeten worden uitgevoerd en het vraagt om veel flexibiliteit. Dat je moet omschakelen klinkt logisch, maar soms merk je niet eens dat je omgeschakeld bent. Je loopt bijvoorbeeld het Agile team binnen en je leest welke 'Conditions of satisfaction' er gelden voor een user story in een sprint en welke taken er nog moeten worden uitgevoerd om als team klaar te zijn. De voortgang is zichtbaar als je in het team zit. De volgende dag ben je een detailtestplan aan het reviewen en testscripts aan het opstellen voor een komende release en ga je morgen de deelacceptatie begeleiden van een onderdeel van de website.

Voorwaarde is dat je veel vrijheid hebt om je eigen rol in te vullen. Je kunt niet full time in een Agile-team zitten en ook niet een volledige applicatie testen. Je rol is breed en dus zijn je taken ook erg breed. Je maakt presentaties, automatische regressietestcases, je begeleidt eindgebruikers, je reviewt user stories maar ook testcases en testplannen, je werkt zelfstandig in een team voor een testcoördinator, soms ben je simpelweg handjes om iets uit te voeren en soms moet je zelfstandig een oplossing uitwerken en invoeren.

Soms merk je dat je nog niet omgeschakeld bent, bijvoorbeeld wanneer je vraagt naar een testrapport over de uitgevoerde testcases in een Agile-team. Grote kans dat je wordt doorverwezen naar het Scrumboard waar de 'gescoorde' user stories hangen. Het moeilijkst is het omgaan met prioriteit van taken van verschillende testmanagers en Scrum-masters. Wat gaat er vóór: de testuitvoer in een Agile-team of de voorbereiding voor de volgende iteraties over twee weken? Mijn oplossing is dan vaak dat ik de testcases die ik uitvoer voor het Agile-team hergebruik voor de andere teams, dus de uitvoer in het ene team is de voorbereiding voor het andere.

Conclusie

Het ideaal is uiteraard dat alle leveranciers perfect samenwerken en een realistische planning afgeven die ook echt wordt gevolgd. Realiteit is dat het een complexe omgeving is om in te werken. Een tester kan de smeerolie zijn om de totale machine soepel te laten draaien. Op het moment dat enablers zoals een centrale releasekalender worden gevolgd en de teams kleine aanpassingen doen aan hun standaard manier van ontwikkelen, dan kan er pas echt kwaliteit worden geleverd. Van de tester wordt daarbij veel gevraagd. Met name moet je heel flexibel en heel breed zijn in de kennis en kunde die je toepast, want alleen zo kun je een optimale bijdrage leveren aan alle teams en aan het totale project. ←