Agile testing: de toekomst?

Businessanalyse: Bedrijfscasus

Klas: TIAO3

Studenten: Ilya Mikhaylov, Collin Verhaeghe

Lector: Martine Van Audenrode

Inhoud

[Inleiding 3](#_Toc94282020)

[Literatuurstudie 4](#_Toc94282021)

[Ontwikkelingsmethodologieën 4](#_Toc94282022)

[Agile toegepast in de praktijk 7](#_Toc94282023)

[Hypothesen 9](#_Toc94282024)

[Vragen voor het interview 9](#_Toc94282025)

[Analyse van het interview 10](#_Toc94282026)

[Liantis 10](#_Toc94282027)

[Zero Friction 11](#_Toc94282028)

[Conclusie 12](#_Toc94282029)

[Referenties 13](#_Toc94282030)

[Bijlagen 14](#_Toc94282031)

[Vragen en antwoorden van de bedrijven 14](#_Toc94282032)

[Liantis 14](#_Toc94282033)

[Zero Friction 15](#_Toc94282034)

# Inleiding

Het testen van software is een integraal onderdeel in het maken van een softwareproduct. Hoe snel het project wordt afgerond en de volledigheid van het product, hangt af van hoe grondig het systeem onderlegd is met testen. Het testen van een softwareproduct in verschillende ontwikkelingsfasen vormt een garantie voor de oplevering van een software product van hoogstaande kwaliteit.

In een steeds versnellende business omgeving is het noodzakelijk om in te kunnen spelen op externe veranderingen. Het traditionele watervalmodel vertoont hierbij tekortkomingen, daar het geen oplossing biedt om contextuele veranderingen tegemoet te komen. Als antwoord op deze noodzaak komt het Agile software development framework steeds meer op de voorgrond te staan.

Agile teams streven ernaar om consistent en regelmatig hoogwaardige nieuwe features te leveren. Het ontwikkelingstempo vereist een adaptieve benadering om de kwaliteit van elke build te waarborgen. Dit staat haak tegenover de traditionele methode daar de testingsfase een afzonderlijke eenheid is die pas van start gaat na afloop van de implementatie van alle functionaliteiten. In organisaties die de traditionele methodes toepassen, zijn vaak testteams aangesteld waarbij ieder lid een sterke achtergrond heeft in Quality Assurance. Bij Agile werken is dit niet van toepassing daar elke functionaliteit herhaaldelijk en incrementeel moet getest en opgeleverd worden.

Het doel van deze paper is om na te gaan welke methodologieën gehanteerd worden door hedendaagse bedrijven en hoe performant deze zijn in contrast met de traditionele benadering. Verder zal beschreven worden hoe deze methodologieën gebruikt worden om frequent releases op de markt te brengen. Mogelijks hebben ze implicaties voor de rolverdeling in applicatie-ontwikkeling-en systeembeheerteams. Om een antwoord te bieden op deze vragen zal eerst de literatuur bestudeerd worden om na te gaan wat de theorie suggereert. Vervolgens zal een interview afgenomen bij twee bedrijven die de Agile methodologie implementeren. In de analyse ervan zal bekeken worden of er al dan niet verschillen bestaan tussen theorie en praktijk.

# Literatuurstudie

## Ontwikkelingsmethodologieën

Voor de bespreking van ontwikkelingsmethodologieën, zal het oudste model – het watervalmodel, eerst beschouwd worden. Vandaag is het nog steeds een van de meest populaire methoden.

Het watervalmodel is zo ontworpen dat men elke fase in het ontwikkelingsproces sequentieel moet doorlopen. Een volgende fase kan dus niet aangevat worden vooraleer de vorige is afgerond en goedgekeurd. Na afloop van elke fase zal het product ervan grondig gedocumenteerd worden. Bovendien is het niet mogelijk om terug te keren naar een stap die voorgaand voltooid is. Deze methode is voornamelijk geschikt voor de ontwikkeling van systemen daar de requirements stabiel zijn. Het biedt immers weinig tot geen mogelijkheid om in te spelen op veranderingen in de eisen (McCormick, 2012)**.**

De volgende stappen geven een bondige beschrijving van de fasen in het watervalmodel. Merk op dat de testing enkel wordt aangevat na afronding van de implementatie fase. De tester heeft dus geen rol in het ontwikkelingsproces in de voorgaande stappen.

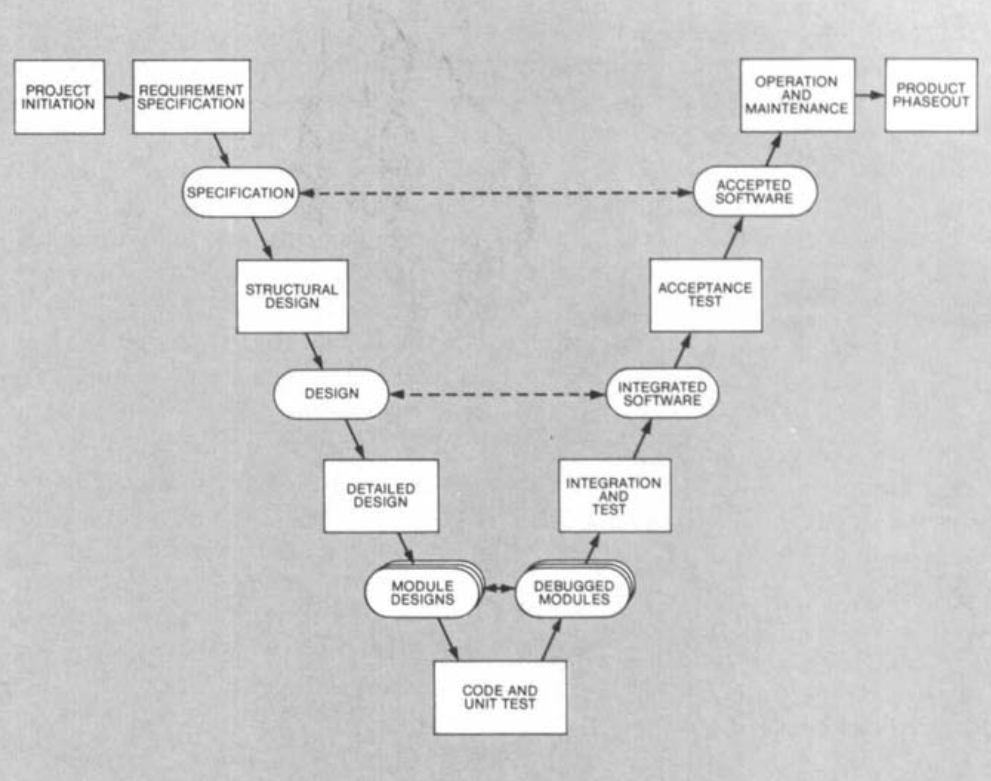
1. Analyse: Een beschrijving van het te ontwikkelen systeemgedrag in een taal die begrijpelijk is voor zowel klanten, alsook ontwerpers. Deze analyse beschrijft ook de zaken die onderworpen moeten worden aan testen de testfase.
2. Ontwerp: De onderdelen die nodig zijn om te systeem te bouwen, worden gedefinieerd. Vaak worden deze onderdelen gespecifieerd als ‘klassen’ met onderlinge relaties die beschrijven hoe ze zullen samenwerken.
3. Implementatie: De eigenlijk broncode wordt georganiseerd en vastgelegd in een taal die te begrijpen is door een computer.
4. Testen: In deze fase vindt het daadwerkelijke testen en valideren plaats. Ze beoogt een antwoord te bieden op de vraag of de software wel werkt zoals verwacht wordt door de klant. Een verscheidenheid aan kwesties worden onderworpen aan deze testen. Voorbeelden hiervan zijn: de interactie tussen systeemcomponenten en de controle of het gebouwde systeem wel voldoet aan de functionele eisen van de klant. Er zijn verschillende manieren om de applicatie te testen. Voorbeelden hiervan zijn prototypering, use test cases, story maps en unit tests. Wanneer fouten naar boven komen tijdens deze fase, is het belangrijk om grondig te documenteren hoe ze tot stand komen. Ze kunnen bijgehouden worden in een ‘issues log’ met een beschrijving en prioriteitsordening. De analist zal vervolgens overwegen wat noodzakelijk is om het probleem te verhelpen, waarna de developer deze oplossing implementeert. Bovendien dient de issues log als een communicatiemiddel om vooruitgang te bespreken met de klant en andere belanghebbenden (Kramer, 2018). Pas wanneer alle kritische fouten zijn opgelost, kan de deployment van de applicatie starten.
5. Deployment: Het nieuwe systeem wordt toegepast in een integratieomgeving, daar het beoogt om nadeloos samen te werken met reeds bestaande systemen. Een best practice omvat het onderwerpen van het systeem aan User Acceptance Testing. Zodanig kan het systeem gevalideerd worden vooraleer het naar de productie-omgeving overgaat.
6. Onderhoud: Na deployment is het nog steeds mogelijk dat fouten optreden bij gebruik van het systeem. Daarom blijft de issues log openstaan om deze problemen te registreren en te verhelpen.

Het watervalmodel is voordelig daar het voor aanvang van het ontwikkelingsproces duidelijk aangeeft wat de eisen van het systeem zijn. Bovendien is elke fase sterk omlijnt en aangeduid met een maximum tijdsbestek. Daarnaast wordt na afloop van elke fase grondige documentatie opgesteld die de kwaliteit van het proces waarborgen.

Daartegenover leidt de toepassing van dit model vaak tot problemen in een omgeving waar veranderende eisen de norm zijn. Omwille van de lineariteit van de methode is het niet mogelijk om terug te keren naar een vorige fase. Problemen worden echter vaak pas ontdekt in latere fases van het ontwikkelingsproces.

Om deze tekortkomingen het hoofd te bieden, werd in 1986 het V-model voorgesteld door Paul Rook (Rook, 1986). Deze methodologie legt in elke stap van het ontwikkelingsproces de link met zijn respectievelijke testfase. Dit in tegenoverstelling tot het klassieke watervalmodel, daar de tester pas zijn rol toegewezen krijgt na afronding van de implementatiefase. Het doel van deze verbinding is om de effectiviteit en de efficiëntie van het ontwikkelingsproces te verhogen, daar het fouten vroegtijdig opspoort wanneer de fase nog niet afgesloten is.

In het V-model wijst elke rechthoek op een ontwikkelingsfase in het proces. Een fase uit zich steeds in een ovaal, het product van de fase. Elk product zal onderworpen worden aan zijn corresponderende test. Beginnend rechtsonder neemt de unit testing phase plaats. Hierbij wordt de functionaliteit van elke component van het systeem geverifieerd. Vervolgens start de integratietestfase. Deze fase beoogt te controleren dat de individuele componenten van het systeem correct functioneren in relatie tot elkaar. Wanneer de functionaliteit van het systeem gegarandeerd is, kan het overgedragen worden aan de klant en zijn productieomgeving tijdens een software acceptatie test. Tot slot is het systeem opnieuw onderhevig aan onderhoudstaken zoals bij het klassieke watervalmodel. Merk op dat de fasen aan de linkerzijde input geven aan de tester. De specificatie van een business analist bepaalt bijvoorbeeld aan welke functionaliteiten het systeem moet voldoen.



**Figuur 1.** Het V-model door P. Rook, 1986, *Software Engineering Journal, 1, p. 7.* Copyright 1986 by Institution of Engineering and Technology (IET).

Zowel het watervalmodel als het V-model doorlopen een strak omlijnde set van fasen, waarbij wederkeren quasi onmogelijk blijkt. Daar testen beter geïntegreerd is in elke fase van het V-model, biedt het geen oplossing om in te spelen op veranderende eisen van de klant. Een antwoord voor dit probleem werd geformuleerd in het Agile model.

Bij de toepassing van de Agile methode trachten ontwikkelaars aan een hoog tempo functionele softwarecomponenten op te leveren na elke sprint. Bijgevolg kan feedback van de klant ingewonnen worden, die dan weer dient om het systeem bij te werken en te verbeteren. Dit staat haaks tegenover het watervalmodel, daar de systeemeisen vooraf vastgelegd worden en hier niet van wordt afgeweken tot aan de oplevering van het complete product. De Agile methode biedt de mogelijkheid om iteratief en incrementeel het product op te bouwen. Een continu behoudt van klantentevredenheid en communicatie staat hierbij op de voorgrond.

Bij Agile development is het aangeraden om het ontwikkelings- en testproces in parallel te laten lopen, gebruikmakend van test-driven-development (TDD). TDD houdt in dat de ontwikkelaar eerst zijn systeemeisen zal definiëren, waarna hij hiervoor unit testen zal ontwikkelen. Vervolgens vindt de eigenlijke implementatie plaats, het coderen van het systeem. De unit testen zullen dan uitgevoerd worden om te verifiëren of de functionaliteit kwaliteitsvol is geïmplementeerd. Tot slot kan een refactoring fase ingelast worden om de architectuur te verbeteren en om bugs te verwerken.

In parallel met elke fase van het ontwikkelingsproces loopt ook een testfase. Het is hierbij belangrijk om op te merken dat de testfase niet expliciet moet uitgevoerd worden door een ‘tester’. De testrol kan deel uitmaken van een ontwikkelaars verantwoordelijkheden. Omgekeerd is het ook mogelijk dat een tester kennis heeft van ontwikkelingsprocedures. Dit versterkt de communicatie tussen beide partijen en helpt om problemen vroegtijdig op te lossen. Multidisciplinaire teams kunnen uitgespeeld worden als troef (Anil, Garg, & Jain, 2014)**.**

Uiteraard vereist grondig testen specifieke kennis van het onderwerp. Daarom kan indien gewenst een onafhankelijk team van testers opgesteld worden die ondersteuning kan bieden voor meerdere ontwikkelingsprojecten tegelijkertijd. Zij zullen niet opnieuw de standaard testen uitvoeren die reeds in gebruik zijn door de ontwikkelaars. De bedoeling is om fouten op te sporen die niet werden gedetecteerd door de eerdere testen. Het voordeel van een onafhankelijk testteam is dat zij de focus kunnen leggen op de perspectief van de klant. Hoe zullen zij het product in gebruik nemen en in welke voldoet het product aan hun eisen? Daarnaast neigen ontwikkelaars te zoeken naar bevestiging voor de volledigheid van hun systeem. Hierdoor kunnen tekortkomingen in de functionaliteiten onopgemerkt blijven. Het onafhankelijk testteam kan deze kwestie behandelen (Pettichord, 2004)

Na elke iteratie zal een stuk code opgeleverd worden die moet voldoen aan een aantal functionele eisen. Doorgaans omvat dit de taak van acceptance testing, die bovendien ook randgevallen van het domein zal detecteren (Parmar, 2016). Buiten functionele code zijn echter een aantal andere testscenarios voorhanden die de kwaliteit van het project kunnen verhogen. Hierbij volgen een aantal voorbeelden:

* Exploratory Testing: Het doel is om nieuwe zaken te ontdekken en om bij te leren van het proces. Tijdens exploratory testing staan testers open voor creatief denken, wat soms kan leiden tot onconventioneel denken. De testen staan niet vast in een script en worden op het moment zelf opgesteld (Hamilton, 2021).
* Interaction Testing: Afzonderlijke componenten zullen uiteindelijk vaak met elkaar moeten samenwerken. Via mocking patronen is het mogelijk om de interactie van deze systemen te controleren (Niederwieser, 2016).
* Load Testing: Het systeem zal hierbij onderworpen worden aan een grote belasting. De resultaten hiervan kunnen onder meer aanwijzen of het systeem al dan niet voldoet aan niet-functionele vereisten.
* Scenario Testing: Belanghebbenden zullen testen of ze een use case kunnen vervolledigen met het gebouwde systeem. Hieruit blijkt snel of er eventuele tekortkomingen zijn in de huidige implementatie.

Om samen te vatten wordt onderstaand een overzicht gegeven van de voornaamste verschillen tussen het Watervalmodel enerzijds en de Agile methodologie anderzijds, gebaseerd op de bevindingen van (Pankaj, 2019).

Agile testen.

* Testen is geen aparte fase en vindt gelijktijdig plaats met de ontwikkeling.
* Testers en ontwikkelaars werken samen.
* Testers nemen deel aan de ontwikkeling van vereisten.
* Na elke iteratie wordt een acceptatietest uitgevoerd.
* Elke iteratie voltooit zijn eigen testen.
* Geen vertragingen tussen coderen en testen.
* Testen is best practice.

Waterval testen

* Testen is een aparte fase. Alle niveaus en soorten testen kunnen pas worden gestart na voltooiing van de ontwikkeling.
* Testers werken apart van ontwikkelaars.
* Testers mogen niet deelnemen aan de requirementsfase.
* Acceptatietesten worden pas aan het einde van het project uitgevoerd.
* Testen kan pas worden uitgevoerd nadat de ontwikkeling is voltooid.
* Meestal vertragingen tussen codering en testen.
* Testen wordt vaak over het hoofd gezien.

## Agile toegepast in de praktijk

Daar de vorige paragraaf duidelijk maakt hoe de Agile methodologie verder bouwt op de tekortkomingen van het Waterval- en V-model, handelt de volgende literatuur eerder over hoe men het toepassen van Agile ervaart in de praktijk.

Onderzoek door Cruzes, Moe en Dyba (2016)die de samenwerking van ontwikkelaars en testers bekijkt, wijst op de efficiëntie in testen die de Agile methodologie teweegbrengt. Ontwikkelaars en testers moeten per noodzaak samenwerken om de doelen van hun softwareontwikkelingsprojecten te bereiken. De projecten zijn van dergelijke complexiteit dat niet kan verwacht worden van een enkel individu dat hij overal kennis van heeft. Men moet beroep kunnen doen op de sterktes van teamleden om obstakels te overkomen. Bovendien zijn in gedistribueerde softwareprojecten de ontwikkeling-en testingteams vaak geografisch verspreid over meerdere locaties. Daarom is regelmatige en gecoördineerde communicatie van uiterst belang. Het toepassen van de Agile methodologie zou de effectiviteit van de communicatie verhogen en leiden tot een efficiënter pad tot het voltooien van sprinteisen door een verhoging in teamproductiviteit.

Ook uit onderzoek van (Begel, 2007) blijken ontwikkelaars en testers meer en beter samen te werken na een implementatie van de Agile methodologie. Deze resultaten blijken uit een anonieme survey bij 492 respondenten, waaronder ontwikkelaars, testers, alsook managers. Voornamelijk dagelijkse scrum meetings zouden leiden tot deze verbeterde samenwerking. Teamleden worden regelmatig op de hoogte gebracht van elkanders activiteiten en het biedt ruimte om problemen naar boven te brengen. Bijgevolg kunnen ze vroeg in het ontwikkelingsproces verholpen worden.

Daarnaast stelt de Agile methodologie de respondenten in staat om op een snel tempo demo releases uit te brengen. Dit maakt het eenvoudiger om vooruitgang te monitoren en bij te sturen indien nodig. Bovendien stelt het de bedrijven in staat hun prototypes voor te leggen aan klanten en feedback in te winnen om hun product te verbeteren in functie van hun wensen. Het vroegtijdig voorleggen van een demobaar product heeft ook een ander positief gevolg daar bugs in de code op een vroeg punt in het ontwikkelingsproces gedetecteerd kunnen worden. Er is dan nog voldoende tijd ter beschikking om fouten in de broncode weg te werken.

Ook vonden de respondenten het mentaal minder belastend om in een Agile framework aan de slag te gaan. Aangezien het ontwerp flexibel is, moet men niet onnodig ‘all in’ gaan op de implementatie van een module die mogelijks niet langer toepasselijk is op de wens van de klant eenmaal die afgewerkt is. Hetzelfde geldt voor de documentatie van het proces. Bij het watervalmodel is de documentatie vaak allesomvattend en uitputtend. Het documenteren van een proces bij de Agile methodologie is beperkt in die zin dat die net volstaat om over te kunnen gaan naar de volgende sprint. Bijgevolg is de Agile methode minder bureaucratisch en kost ze vaak minder.

Uit de resultaten blijkt echter ook dat niet alles rooskleurig is voor Agile. Een van de grootste bekommeringen is dat Agile ontwikkelingsmethodologieën niet schaalbaar zouden zijn voor grotere projecten. Hoe groter de omvang van het project, hoe moeilijker het immers wordt om dynamisch van ontwerp te veranderen. Een grotere hoeveelheid werk zou verloren gaan.

Daarnaast ervaren de respondenten regelmatige scrum meetings als inefficiënt. Dan voornamelijk wanneer het team weinig ervaring heeft of wanneer de grootte van de groep groter is dan 10 personen. Een andere bekommering over scrum meetings is dat het management deze zouden gebruiken om teams de micro-managen, in die zin dat zij elk gespendeerd werkuur gerechtvaardigd willen zien.

Uit de antwoorden van de management respondenten blijkt daarnaast dat men niet gerust is in de verwachtte adoptie van het upper-management. Daar is de Agile methodologie mogelijks niet gekend, maar zullen vooruitgangsrapporten wel nog altijd geëist worden van de onderliggende werknemers. Bij Agile is het echter moeilijk om deze vooruitgang te documenteren en voor te leggen aan hun oversten. Een voordeel van het Watervalmodel is dat de scope van het project strak omlijnt is in functie van tijd. Die is minder het geval bij Agile daar het moeilijk is om te voorspellen wanneer het product afgewerkt zal zijn naar de wensen van de klant.

Ondanks deze bekommeringen blijkt echter uit een studie van (Salo & Abrahamsson, 2008) dat de acceptatie van de Agile ontwikkelingsmethodologie toeneemt. Voornamelijk in grote en gedistribueerde omgevingen komt Agile steeds meer aan de orde. Dit blijkt uit een survey bij 35 projecten in de software industrie, waaronder 3 in België. 77% van de respondenten oordeelt dat het toepassen van scrum een meerwaarde biedt voor het softwareontwikkelingsproject.

Samengevat blijkt dat de Agile methodologie niet komt zonder zwaktes, maar wel een antwoord kan bieden op een verscheidenheid van tekortkomingen in het watervalmodel. Het wordt ook positief ervaren door de meerderheid van de teams die het trachten toe te passen in hun ontwikkelingsproces.

## Hypothesen

* In de literatuur werd bevonden dat binnen Agile het testen quasi parallel verloopt met de applicatieontwikkeling. Bijgevolg verwachten we dat organisaties die Agile toepassen de nieuwe software vroegtijdig testen om problemen op te sporen en de kwaliteit van hun product te garanderen
* De agile methodologie suggereert een afwezigheid van een sterke scheiding tussen testers en ontwikkelaars. Het bevorderen van communicatie tussen testers en ontwikkelaars zou de kwaliteit van het eindproduct bevorderen. Bovendien zijn testers en ontwikkelaars beter op de hoogte van elkanders werk dankzij de dagelijkse stand-up meetings. Bijgevolg verwachten we dat de organisaties geen strikte scheiding hebben tussen testers en ontwikkelaars en dat diegene met de testverantwoordelijkheid betrokken is bij het volledige verloop van het ontwikkelingsproces. En dus niet enkel nadat de ontwikkelingsfase is afgerond zoals dat het geval is bij de watervalmethode.

In wat volgt zullen twee organisaties onder de loep genomen worden om te kijken hoe zij frequente releases realiseren en hoe zij hun ontwikkelings- en testteams indelen in functie ervan. Bij elk bedrijf werd een interview afgenomen om een beter beeld te scheppen van hun situatie. Merk op dat de bedrijven sterk verschillen in grootte. Liantis heeft momenteel ca. 1800 medewerkers, daar Zero Friction er slechts 15 heeft. Nadien zullen hun ervaringen getoetst worden aan de bevindingen uit de literatuur om te kijken in welke mate er overeenstemming is.

# Vragen voor het interview

De volgende vragen zullen gesteld worden aan twee bedrijven om beter inzicht te verwerven in de manier waarop ze het implementeren van de agile methodologie ervaren:

* Welk soort software ontwikkelt het bedrijf?
* Hoe frequent zijn de software releases van het bedrijf?
* Welke testen worden gehanteerd vooraleer de software wordt opgeleverd?
* Wie in de organisatie is verantwoordelijk voor het schrijven van testen?

Afhankelijk van de antwoorden werden enkele vragen verder uitgediept om een antwoord te bieden op de hypothesen.

# Analyse van het interview

## Liantis

Voor hun interne behoeften gebruikt het Liantis-team de moderne Agile-methodologie. Een sprint duurt twee weken, waarbij het team werkt aan een door de klant aangevraagde feature. Men gebruikt er de meest populaire applicatie-stack voor enterprise development, namelijk front-end development met Angular en backend development in Java. De communicatie ertussen wordt mogelijk gemaakt via een RESTful API. Deze klassieke aanpak vergemakkelijkt de implementatie van nieuwe functionaliteiten op aanvraag, maar het testen en bouwen van het project vereist extra stappen, waaronder front-end testen van de klantzijde.

Om tests uit te voeren, gebruikt het bedrijf "Junit" voor backend development. Junit is een testbibliotheek voor Java-software. Het idee is om tests te schrijven voor elke niet-triviale functie of methode. Hiermee kan men snel controleren of de volgende wijzigingen in de code al dan niet leiden tot regressie, dat wil zeggen tot het verschijnen van fouten in de reeds geteste delen van het programma. Het vergemakkelijkt ook de detectie en eliminatie van dergelijke fouten. Het "Selenium Framework" - dat door het bedrijf wordt gebruikt voor front-end prestatietests - is een soort niet-functionele test die meet hoe webapplicaties en webpagina's reageren wanneer ze onderworpen worden aan verschillende belastingniveaus. Een voorbeeld hiervan is het aantal verzoeken tot een bepaalde pagina op een specifiek ogenblik. Iedereen die iets routinematig herhaalt, probeert het zoveel mogelijk te automatiseren, dus wordt Selenium gebruikt om het testen van webapplicaties te automatiseren. Het bedrijf gebruikt "SonarQube" voor algemene dekking. SonarQube is een platform voor codeanalyse en kwaliteitsbeheer. Het past mooi in een reeks processen, maakt continue code-analyse mogelijk en werkt met meerdere talen met behulp van een plug-in-systeem. Met SonarQube kan men de huidige staat van de codebase analyseren op verschillende bugs/kwetsbaarheden. Het maakt het ook mogelijk om het aantal van deze bugs/kwetsbaarheden te verminderen en in de toekomst, dankzij continue analyse, de kwaliteit van de code te verbeteren.

Liantis gebruikt verschillende manieren voor het testen van hun applicaties. Zij beweren aan Test-Driven Development te doen voor de backend. Concreet betekent dit dat een test geschreven wordt voor de implementatie van een bepaalde functionaliteit. Dit garandeert op formele wijze dat de logica doet wat het beweert te doen zoals gedefinieerd in de test. Deze manier van werken wordt door Liantis toegepast op alle lagen in het backend project. Indien de Unit-testen slagen worden verder ook integratie-tests geïmplementeerd om na te gaan of de communicatie tussen de verschillende domeinlagen correct verloopt.

Voor de front-end worden naast Unittests ook automatische testen toegepast die het gebruiksgedrag van hun klanten nabootst. Zodanig kunnen fouten vroegtijdig opgespoord en afgehandeld worden. De automatische testen worden opgezet door een specifiek daartoe aangewezen testpersoon.

Concreet trachten Liantis elk maand een release te voorzien voor hun legacy-applicaties. Het tempo ligt iets hoger bij moderne applicaties, daar de periode twee weken bedraagt. Ze trachten aan het einde van iedere sprint als onafhankelijk team de nieuwe features te deployen. Liantis pushen hun code naar een development branch daar de testen nogmaals gedraaid worden. Indien de testcoverage een bepaald percentage bedraagt, kan de code gedeployed worden. De effectieve deploy vereist echter de goedkeuring van het QA-team. Dit team bestaat onder meer uit de hierboven vernoemde aangewezen tester en een analist.

## Zero Friction

Zero Friction is een Startup in het Gentse die een facturatieplatform aanbiedt aan energieleveranciers. Als jong bedrijf zijn ze momenteel met 15 werknemers, maar dit is sterk onderhevig aan verandering daar ze openstaan voor de werving van nieuw personeel.

Het bedrijf houdt zich momenteel bezig met de ontwikkeling van een ERP systeem die geschreven is in .NET voor de backend en React voor de frontend. Daarnaast trachten ze hun klanten (de verbruikers) ook te voorzien van een portaal die customizable is aan hun eigen noden. Deze is geschreven in .NET voor de backend en Blazor voor de frontend.

Recentelijk is Zero Friction begonnen met toepassen van een Kanban systeem, waarbij de productmanager en ontwerper tweewekelijks nieuwe functionaliteiten aanbrengen die geïmplementeerd moeten worden in de bijhorende applicatie. In samenspraak met het ontwikkelingsteam wordt dan overwogen of dit een haalbaar project is. Tevens kunnen ook opmerkingen gedeeld worden over wat er al dan niet goed loopt bij de ontwikkeling van vorige features. De bedoeling is om op de laatste dag van elke sprint nieuwe features te releasen voor hun klanten. De ontwikkelaars van het bedrijf geven aan enige druk te ervaren om features tijdelijk uit te brengen. Zij kaarten zelf aan dat ze dit belangrijk vinden om grotere klanten aan te trekken.

Naast een toekomstgericht perspectief te overwegen, blikken zij na elke sprint ook terug over hoe het proces verloopt in een retrospective. Zero Friction halen aan dat ze zeer open zijn om van werkwijze te veranderen indien dit zich voordelig kan uitspelen voor de doelen van het bedrijf.

Om het ontwikkelingsproces te monitoren maakt Zero Friction gebruik van Azure Devops. Daarop worden alle sprints, tickets en andere problemen bijgehouden. Daarnaast is er een QA omgeving voorzien waarop alle functionaliteit eerst grondig getest wordt.

Net zoals bij Liantis wordt de backend logica onderworpen aan Unittests om de correctheid ervan te garanderen. Deze Unittesten worden doorgaans opgeleverd aan het ontwikkelingsteam bij aanvang van een sprint, samen met bijhorende mock-ups. Bij het committen van code naar de betreffende repository worden online nogmaals testen gedraaid om te verzekeren dat er geen fouten opgetreden zijn. Voor de frontend kon Zero Friction weinig informatie geven aangezien ze momenteel in een experimentele fase zitten om ook structuur te bieden aan testen voor frontend logica. Ze geven echter wel aan te experimenteren met Jest om Unit testing te verwezenlijken. Er werd de afgelopen sprints hard ingezet op het herstructureren en centraliseren van logica om het testen ervan eenvoudiger te maken.

Buiten Unittesten voorzien de ontwikkelaars van Zero Friction voor afloop van elke sprint een dagen de tijd om end-to-end testing te doen. Zodanig kunnen problemen opgespoord worden met voldoende resterende tijd om een oplossing te kunnen bieden tegen de afloop van de sprint waarin deze zijn opgetreden.

Algemeen ligt de verantwoordelijkheid van testen bij de ontwikkelaars zelf. Vaak krijg een ontwikkelaar de taak om zowel de test als de implementatie van een bepaalde functionaliteit te voorzien. Zero Friction lieten weten dat ze momenteel de nood voelen om een analyst aan te werven om de te realiseren functionele en niet-functionele requirements van een bepaalde sprint aan te scherpen. Momenteel is dit vaak nog open voor eigen interpretatie, wat het moeilijk maakt om verwachtingen overeen te stemmen. Daarnaast overwegen ze ook om een aangewezen test persoon aan te nemen om een deel van de testverantwoordelijkheid over te nemen van de ontwikkelaars.

# Conclusie

Uit de literatuur bleek dat de Agile ontwikkelingsmethodologie het toelaat voor bedrijven om testen parallel te laten verlopen met de implementatie van domeinlogica. Bij zowel Liantis als Zero Friction is dit het geval. Testen wordt niet uitgesteld tot na de afloop van de implementatiefase. Het is zelfs een vereiste om de correctheid van domeinlogica te bewijzen vooraleer deze functionaliteiten kunnen overgedragen worden tot productie. Merk op dat beide organisaties inzetten op Test Driven Development en dus Unit-tests ontwikkelen vooraleer de eigenlijke implementatie van de functionaliteit wordt aangevat.

Daarnaast trachten beide bedrijven regelmatig nieuwe functionaliteiten op te leveren. Deze releases vinden normaliter plaats elke twee weken. Dit zou niet mogelijk zijn indien zij gebruik maakten van de watervalmethode, daar alle functionaliteiten van het softwarepakket op het einde van het ontwikkelingsproces worden opgeleverd.

De literatuur suggereert verder dat het toepassen van de Agile methodologie de samenwerking tussen tester en ontwikkelaar zou bevorderen omwille van dagelijkse stand-up meetings. Uit de interviews is het onmogelijk om te concluderen of ze al dan niet deze stand-up meetings uitvoeren. Zero Friction zet bijvoorbeeld wel in op een meeting bij aanvang van elke sprint om obstakels bij het ontwikkelingstraject weg te werken en om beter inzicht te verwerven in de activiteiten waartoe hun teamleden betrokken zijn.

Ook werd wel duidelijk dat beide organisaties overlap vertonen in de verantwoordelijkheden die oorspronkelijk in het watervalmodel volledig gescheiden zouden zijn. De ontwikkelaar krijgt bij zowel Liantis als Zero Friction een deel van de testverantwoordelijkheid toegewezen. Liantis voorziet echter een testpersoon die de manuele en automatische tests opvolgt vooraleer de functionaliteit tot productie wordt overgedragen. Dit is niet het geval bij Zero Friction, maar zij voelen wel aan dat dit in de toekomst een noodzaak zou kunnen worden indien de scope van hun projecten toeneemt.

In ieder geval is de meerwaarde van vroegtijdig testen duidelijk. Fouten kunnen opgespoord worden vooraleer de functionaliteiten de klant bereiken. Er wordt ook voldoende tijd voorzien in sprints om de ontwikkeling van oplossingen voor deze problemen mogelijk te maken.

In vervolgonderzoek kan het interessant zijn om een interview af te nemen bij een organisatie die meer geïnvesteerd is in een testteam. Bijgevolg kan nagegaan worden hoe de relatie wordt opgebouwd en onderhouden tussen testers en ontwikkelaars. Ook moet opgemerkt worden dat de interviews werden afgenomen bij ontwikkelaars van de desbetreffende organisaties. Zij hebben er bijgevolg belang bij om niet uit te weiden over aspecten betreffende bedrijfspolitiek. Mogelijks zijn zij niet tevreden met het status quo, maar hebben alsnog weinig inzeg hierop.

Daarnaast moet kan opgemerkt worden dat beide Zero Friction’s bedrijfsmodel berust op Software as a Service (SaaS). Men kan veronderstellen dat de Agile ontwikkelingsmethode hiervoor uiterst geschikt is, daar het regelmatig opleveren van software ondersteunt en de ontwikkelaar regelmatig voorziet in klantenfeedback. Maar mogelijks kan de Agile methode ook voordelig zijn in organisaties die niet berusten op SaaS.

# Referenties

Anil, A., Garg, N., & Jain, A. (2014). Quality Assurance for Product Developmentusing Agile. *International Conference on Reliability, Optimization and Information Technology*.

Begel, A. (2007). Usage and Perceptions of Agile Software Development in an Industrial Context: An Exploratory Study.

Cruzes, D. S., Moe, N. B., & Dyba, T. (2016). *Communication between Developers and Testers in Distributed Continuous Agile Testing*. Retrieved from IEEE Xplore.

Hamilton, T. (2021). What is Exploratory Testing? Techniques with Examples. *Guru99*.

Kramer, M. (2018). BEST PRACTICES IN SYSTEMS DEVELOPMENTLIFECYCLE: AN ANALYSES BASED ON THEWATERFALL MODEL. *Review of Business & Finance Studies*.

McCormick, M. (2012). Waterfall vs. Agile Methodology. *MPCS, Inc.*

Niederwieser, P. (2016). *The Spock Framework Team*. Retrieved from https://spockframework.org/spock/docs/1.1/interaction\_based\_testing.html.

Pankaj, P. (2019). *Difference between Agile Testing and Waterfall Testing*. Retrieved from Geek for geeks.

Parmar, D. (2016). *Exploratory testing*. Retrieved from https://www.atlassian.com/.

Pettichord, B. (2004). Agile Testing Challenges. *Thought Works*.

Rook, P. (1986). Controlling software projects. In Software Engineering Journal (Vol. 1, Issue 1, p. 7). Institution of Engineering and Technology (IET). https://doi.org/10.1049/sej.1986.0003

Salo, O., & Abrahamsson, P. (2008). Agile methods in European embedded software development organisations: A survey on the actual use and usefulness of Extreme Programming and Scrum.

# Bijlagen

## Vragen en antwoorden van de bedrijven

### Liantis



*Als sociaal verzekeringsfonds vormt Liantis de schakel tussen zelfstandigen en de overheid. Ze zorgen voor aansluiting, berekenen de sociale bijdrage en helpen mensen om hun sociale zekerheid te optimaliseren. Ze bieden oplossingen voor salarisadministratie, kinderbijslag- en sociale zekerheidsfondsen, werving en selectie, preventie en bescherming: het aanbod van Liantis bestrijkt een brede waaier van professionele diensten.*

**Welke soort software ontwikkelt Liantis?**

“In onze team ontwikkelen we voornamelijk web-toepassingen voor intern gebruik die bepaalde bedrijfsprocessen automatiseren. De meeste toepassingen volgen een micrposervices architectuur, met uitzondering van enkele legacy applicaties die nog als monolithen geschreven zijn. De back-ends zijn voornamelijk in Java 8 en 11 geschreven met gebruik van de Spring Framework en Springboot voor de nieuwere toepassingen. De front-ends staan als aparte app's gedeployed die dan communiceren met de back-ends via REST en ze zijn ontwikkeld in Angular.”

**Welke soort testen hanteert het bedrijf?**

“Voor onze testen gebruiken we enerzijnds Unit tests als Automatische tests (ook wel “automagische” genoemd bij ons). Unit tests schrijven we als deel van onze Test-Driven Development methodiek om ervoor te zorgen dat we een zekere formele manier hebben om te controleren dat nieuw opgeleverde features aan bepaalde acceptatiecriteria voldoen alsook om regressies te voorkomen. Hierbij testen we alle lagen van de back-end app–controllers, services, repositories, utility classes, mappers enzovoort alsook integratietests (testen die de volledige flow van rest end-point → business logica → persistence layer aftoetst). Hiervoor gebruiken we Junit framework. Voor de front-ends worden er ook unit tests geschreven. Als we onze features pushen naar de build server worden die unit tests ook uitgevoerd en als er voldoende coverage is en de configuratie in orde is wordt de build successvol gedeployed. Daarnaast draaien er ook automatische tests op regelmatige basis die dan in feite het menselijke gebruik van onze toepassingen proberen na te botsen aan de front-end via de browser om ervoor te zorgen dat er geen regressies optreden in zowel de bestaande als de recent ontwikkelde functionliateit. Onze automatische tests zijn voornamelijk in Selenium geschreven. Voor test coverage gebruiken we SonarQube. We hebben ook een dedicated tester die zich volledig bezig houdt met zowel automatische als manuele tests.”

**Hoe frequent zijn de software releases van het bedrijf?**

“Binnen Liantis werken we momenteel met zowel maandelijkse releases (voornamelijk voor legacy-toepassingen) en releases die we op zelfstandige basis als team doen op het einde van de sprint (iedere 2 weken). Voor we nieuwe features/bugfixes releasen naar productie, worden ze eerst op de ontwikkelomgeving gedeployed en getest. Als de build slaagt, de unit en automatische tests slagen met voldoende coverage dan pas wordt die feature branch gemerged met de ontwikkel branch. Vervolgens moet deze feature nog goedgekeurd worden door QA (onze tester of analyst) en dan pas op het einde van de sprint mag die gereleased worden naar productie. De build automation tool die we gebruiken om dit te realiseren is Jenkins.”

### Zero Friction



*Wij bieden een facturatieplatform aan (ERP) voor energieleveranciers. Meer specifiek richten wij ons op de noden van warmteleveranciers. Onze diensten zijn in de vorm van SaaS en BPO (onze klanten gebruiken de software via licentie of wij beheren zelf de gegevens).*

* *Op dit moment hebben wij 2 grote softwareprojecten:*
* *Onze ERP (.Net backend en React frontend).*

*Een klantenportaal voor de eindgebruikers (verbruikers) van onze klanten. Dit portaal is customizable volgens de look en feel van onze klanten. (.Net backend en Blazor frontend)*

**Hoe frequent zijn de software releases van het bedrijf?**

“We proberen te werken in sprints van 2 weken volgens een Kanban systeem. De bedoeling is om op de laatste dag van elke sprint een release te doen naar productie. Dit gebeurt enkel indien er geen significante issues opduiken na user testing. We maken gebruik van Azure Devops om de sprints, tickets, issues te monitoren zowel voor planning als voor statistieken (waar ging het fout).

We maken gebruik van een QA omgeving waarop alles eerst wordt getest (door een fysieke gebruiker, meestal teamleden) alvorens de code naar productie gaat.

Bij elke push via git wordt de nieuwe code gescand op fouten en runnen de unit tests via de pipeline.

Gedurende een sprint wordt een analyse en design (mockups) opgeleverd door onze productmanager en designer. Deze worden besproken samen met de developers (laat onze architectuur een bepaalde manier van werken in de mockups toe, of kunnen we een nieuwe manier tijdig implementeren naast de nieuwe features) en indien nodig bijgestuurd.

Bij de start van een nieuwe sprint nemen we deze mockups er opnieuw bij tijdens de taakverdeling.”

**Hoe ervaren jullie het gebruik van sprints bij Zero Friction?**

**“**We kunnen op deze manier sneller inspelen op eventuele feedback van onze klanten. Hoe kleiner de releases hoe evidenter het is om nog zaken bij te sturen in de volgende sprint indien nodig. Een zekere werkdruk is er sowieso aangezien we als startup bepaalde functionaliteiten moeten kunnen ondersteunen om grotere klanten aan te trekken. Dit varieert echter van tijd tot tijd.”

**Worden momenteel ook andere ontwikkelingsstrategieën overwogen binnen de organisatie?**

“Dit is een continu proces. Er loopt altijd wel een experiment om onze processen verder te stroomlijnen. Aan het einde van elke sprint houden we ook een retrospectieve om de manier van werken bij te sturen. Voorbeeld: actueel zijn we bezig met een nieuw state management systeem door te voeren in de front-end (minder gevoelig voor bugs).”

**Welke soort testen hanteert het bedrijf?**

“We maken gebruik van unit testen in de backend, meestal opgeleverd samen met de nieuwe functionaliteit. Indien het om bugs gaat wordt dit scenario ook als test toegevoegd. In de frontend zijn we nu in een experimentele fase voor testen, het aantal effectief gebruikte testen is hier momenteel beperkt. Naar het einde van elke sprint toe wordt er 2 à 3 dagen getest door elk lid van het (ontwikkel)team. Dit zijn end to end usertesten. Indien er nog belangrijke problemen opduiken kunnen deze dan nog tijdig worden opgelost.”

**Wat bedoelt u precies met een experimentele fase?**

“We hebben al experimenten gedaan met Cypress (UI testing) en met Jest unit testen. De tweede gaan we op termijn uitbreiden omdat testen door ons nieuw state management veel evidenter zal worden. De logica is nu meer gecentraliseerd en gestructureerd in testbare klassen.”

**Wie is aangesteld in de organisatie om testen te schrijven. Verschilt dit per type test en bestaat er overlap tussen testers en developers?**

“De verantwoordelijkheid voor testen ligt bij de developers. Vaak worden functionaliteit en test door dezelfde persoon geschreven. We hebben momenteel geen tester in dienst, dit zou op termijn kunnen veranderen”