Bachelorscriptie
Informatiekunde

Radboud Universiteit

Het gebruik van architectuurconcepten bij de ontwikkeling van web/mobiele applicaties

Auteur:
Paul Huberts
s0814873

Inhoudelijk begeleider:
Dr. Stijn Hoppenbrouwers
stijnh@cs.ru.nl

Tweede lezer:
Dr. Tom Claassen
tomc@cs.ru.nl

26-6-2016

Samenvatting

In deze scriptie wordt onderzocht of er concepten uit de digitale architectuur gebruikt worden bij het ontwikkelen van mobiele en/of webapplicaties. Het gebruik van concepten uit de digitale architectuur kan helpen om het ontwikkelproces goed en gestructureerd te laten verlopen. Het is een kwalitatief onderzoek waarbij data is verzameld door middel van het afnemen van interviews. Uit deze interviews is gebleken dat er bij organisaties met maximaal enkele tientallen werknemers slechts minimaal gebruik wordt gemaakt van concepten uit de digitale architectuur. Bij organisaties met enkele honderden werknemers wordt gebruik gemaakt van software architecten om het product te ontwerpen.

Inhoud

Hoofdstuk 1

Inleiding

Tussen het maken van fysieke producten zoals een kast of een huis en het maken van software zijn veel overeenkomsten te vinden. Zo maak je eerst een plan van wat je wilt gaan maken voordat je daadwerkelijk begint. Bij het maken van kleine producten wordt deze stap nogal eens overgeslagen, het gevoel heerst dat het omslachtig is. Over het maken van iets relatief eenvoudigs als een hondenhok wordt snel gedacht dat hier geen bouwtekening voor nodig is, daarom wordt dit ook vaak niet gedaan. Het overslaan van het maken van een ontwerp vergroot het risico dat er onderdelen vergeten worden.

Er kunnen ook nadelen zitten aan het maken van een plan: er wordt zo goed over een product nagedacht dat er te veel regels en teveel eisen aan verbonden worden. Om ervoor te zorgen dat het verkeer bij Roermond niet meer door de stad hoeft is er besloten om een tunnel aan te leggen. In de tijd dat deze tunnel gebouwd werd is er een nieuwe tunnelwet aangenomen. Deze wet had als gevolg strengere veiligheidseisen waardoor vergeleken met het verleden tunnels sneller afgesloten worden. Nadat de bouw van de tunnel voltooid was, is deze in de eerste 10 dagen 11 keer afgesloten [4].

Hetzelfde kan ook gebeuren bij grote IT projecten. Er worden dan zoveel eisen aan het product gesteld dat dit bijna niet in een keer goed te doen is. Om ervoor te zorgen dat projecten overzichtelijk zijn en blijven wordt er in de IT net als bij de fysieke producten gebruik gemaakt van architectuur. Op deze manier kan er systematisch over een softwareproduct nagedacht worden zonder dat het onoverzichtelijk wordt of dat er in details verzand wordt. Dit gaat dan ook vaak samen met het gebruik van plaatjes. De informatie in een plaatje is in veel gevallen sneller en makkelijker te begrijpen dan een aantal pagina's tekst. Het is daarom ook te begrijpen waarom figuren bij grote projecten worden gebruikt, het is namelijk onmogelijk om anders het overzicht te behouden.

Het is niet duidelijk hoe er met architectuur omgegaan wordt bij mobiele en webapplicaties. Door de explosieve groei van het aantal mobiele en

webapplicaties worden er veel meer relatief kleine programma's geschreven. Een groot project zoals de Mozilla Firefox browser bestaat uit bijna 10 miljoen regels code terwijl een mobiele applicatie gemiddeld uit ongeveer 40.000 regels code bestaat. Zo zijn er inmiddels 1.6 miljoen applicaties verkrijgbaar in de Google Play Store en 1.5 miljoen in de Apple App Store [2].

Het ontwikkelen van mobiele en webapplicaties gebeurt steeds sneller, dit komt doordat de verschillen tussen ontwerp, ontwikkeling en gebruik steeds kleiner worden. Volgens Fugetta [12] komt dit mede doordat het ontwikkelen sneller moet dan voorheen. Door deze fases samen te voegen wordt de tijdspanne tussen ontwikkelen en de ingebruikname verkort. Doordat de infrastructuur rondom de software blijft veranderen is het van belang dat er aanpassingen aan de software gedaan worden om een correcte werking te garanderen. Door de tijdsbesparing als, gevolg van het samenvoegen van de verschillende stappen die doorlopen worden tijdens het ontwikkelen van een programma, bestaat het gevaar dat nuttige stappen overgeslagen worden.

Ondanks dat er een steeds kortere ontwikkeltijd is voor mobiele en webapplicaties zijn er genoeg redenen om van architectuur gebruik te maken. Bijvoorbeeld om goed om te kunnen gaan met veranderende infrastructuur. Het is alleen niet duidelijk hoe er in de praktijk gewerkt wordt.

In deze scriptie wordt geprobeerd antwoord te vinden op de vraag: "Worden er bij het ontwikkelen van Webapplicaties/Mobiele applicaties concepten uit digitale architectuur gebruikt?

Deze vraag wordt in een aantal stappen beantwoord. Eerst wordt er gezocht naar bestaande architectuurconcepten die gebruikt worden bij het ontwikkelen van software, het gaat hierbij over software ontwikkeling in het algemeen. Vervolgens wordt er aan de hand van deze concepten een interviewschema opgesteld. De interviews worden gecodeerd. Deze codering maakt het mogelijk om op een systematische wijze de ontwikkelcyclus bij verschillende organisaties te vergelijken. Deze vergelijking maakt het mogelijk om na te gaan of de eerder gevonden concepten voorkomen in de praktijk of dat er een andere werkwijze gebruikt wordt.

Hoofdstuk 2

Theoretisch Kader

2.1 Wat zijn mobiele/webapplicaties?

Er zijn veel mobiele en webapplicaties beschikbaar en er zijn veel bedrijven die dergelijke producten ontwikkelen.Er zijn twee type applicaties die ontwikkeld kunnen worden. Dit zijn de native applicaties, deze draaien op het apparaat zelf. Of dit zijn webapplicaties, deze draaien door middel van de browser. Er wordt vaak gedacht dat native applicaties sneller zijn, het verschil is echter alleen duidelijk bij games of beeldbewerkings-programma's [8]. Omdat dit niet meer het geval is worden er steeds meer webapplicaties ontwikkeld.

Hokannen [14] vertelt dat tijdens haar onderzoek duidelijk werd dat onder start-ups de gedachte heerst dat alleen een uniek en/of innoverend product leveren niet genoeg is. Naast het belang van innovatie en uniciteit is een goede gebruikerservaring van groot belang voor het succes van het product. Dit onderzoek laat dus zien dat er verschillende aspecten belangrijk zijn bij het succes van een product. Het gebruik van modellen is uiterst geschikt om te zorgen dat er controle is over de verschillende aspecten in de ontwerpfase.

2.2 Het ontwikkelen van mobiele/webapplicaties

Bij het ontwikkelen van applicaties zijn er veel factoren die van invloed zijn op het ontwikkelproces. Dit kunnen onder andere veranderende eisen van de opdrachtgever, complexiteit van de opdracht of gebruikte ontwikkelmethode zijn. Als de applicatie door een aantal ontwikkelaars geschreven wordt, dan moeten alle fases door deze ontwikkelaars doorlopen worden. Vanaf het ontwerp van de applicatie tot aan het testen. Bij het ontwerpen wordt gebruik gemaakt van modellen om vooraf een idee van het eindproduct te krijgen. Bij het testen kan gebruik gemaakt worden van unittests. Zoals Joorabchi [17] aangeeft zijn deze tests bij mobiele applicaties niet perfect.

Dit omdat het niet duidelijk is in welke omgeving de applicatie gebruikt gaat worden. Een voorbeeld hiervan is dat de input van een bewegingssensor niet direct te testen is in de applicatie door middel van een dergelijke test. Dit laat zien dat de ontwikkeling die plaatsvindt op technologisch vlak, zoals de komst van de smartphone, er voor zorgt dat niet alle beschikbare middelen om een applicatie te ontwikkelen even bruikbaar blijven. Voor applicaties die in een statische omgeving gebruikt worden zijn dit soort tests nog wel bruikbaar. Veel kleine bedrijven hebben een gebrek aan beschikbare middelen, of dit geld, mankracht of tijd is. In deze gevallen wordt er de keuze gemaakt om het product met beperkte functionaliteiten op de markt te brengen. Hierbij ligt wel de nadruk op een goede gebruikersinterface. Als de gebruiker te veel moet nadenken sluit deze de applicatie weer af. [13]

Er zijn verschillende ontwikkelmethodes waar gebruik van gemaakt kan worden, elke heeft zijn voor en nadelen. Drie van deze ontwikkelmethodes zijn, Waterval, Rational Unified Process en Agile.

2.2.1 Gebruik van Waterval

Waterval is een van de oudste manieren die er is om software te ontwikkelen. Het ontwikkelen volgens de waterval ontwikkelmethode gaat door vijf stappen te doorlopen (figuur 2.1). In de eerste stap wordt er nagedacht

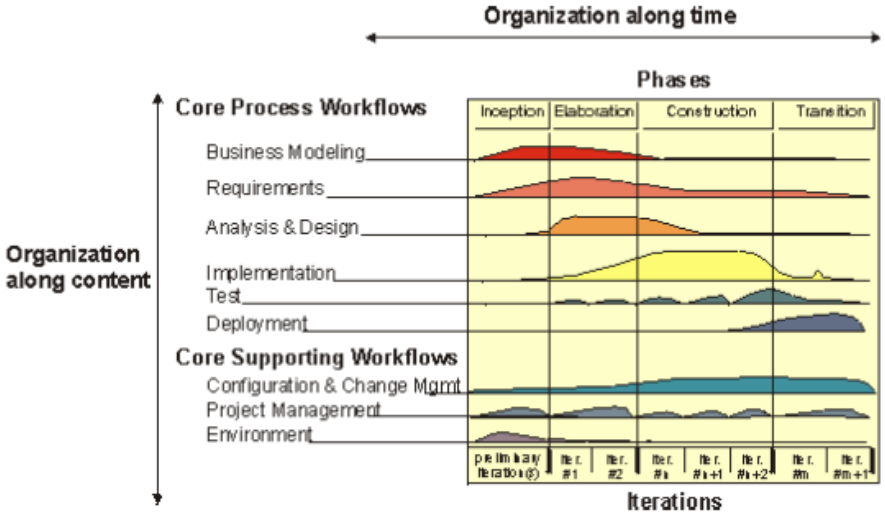
Figuur 2.1: Waterval ontwikkel cyclus

over het product dat ontwikkeld gaat worden. Hierbij worden de vereisten van het product geanalyseerd en vastgelegd. Daarna wordt er in de tweede stap een ontwerp gemaakt. Dit is zowel het technische(hoe zit het technisch in elkaar) als het functionele ontwerp(hoe werkt het en welke handelingen moeten gedaan worden om het gewenste resultaat te krijgen). Als dit klaar is wordt in stap drie de software geschreven. Vervolgens wordt de software getest in stap vier. In stap vijf wordt de software opgeleverd en is deze klaar voor gebruik. Bij het gebruik van de waterval ontwikkelmethode kan er pas met de volgende stap begonnen worden als de vorige afgerond is [16].

2.2.2 Gebruik van Rational Unified Process

Het Rational Unified Process (RUP) is ontwikkeld in de jaren '90 en was bedoeld om een aantal nadelen van het ontwikkelen met bijvoorbeeld Waterval weg te nemen. Deze nadelen waren onder andere het gebrek aan

vroeg risicomanagement en geen versiebeheer. Ondanks dat het gebruik van iteratieve ontwikkelmethodes al sinds de midden jaren '50 plaatsvond, is er door Boehm een spiraal model voorgesteld waar later RUP op is gebaseerd [21], een iteratieve ontwikkelmethode. RUP bestaat uit vier fases waarbij een fase uit meerdere iteraties kan bestaan (zie figuur 2.2). De eerste fase is het ontstaan, hier worden de kosten, risico's en architectuur geïdentificeerd. De tweede is uitwerking, hier worden de vereisten in detail vastgelegd en de architectuur gevalideerd. De derde is bouw, hier wordt software ontwikkeld, getest en gedocumenteerd. De laatste is transitie, hier wordt het systeem getest, de software door gebruikers getest, aangepast en uitgerold [26].

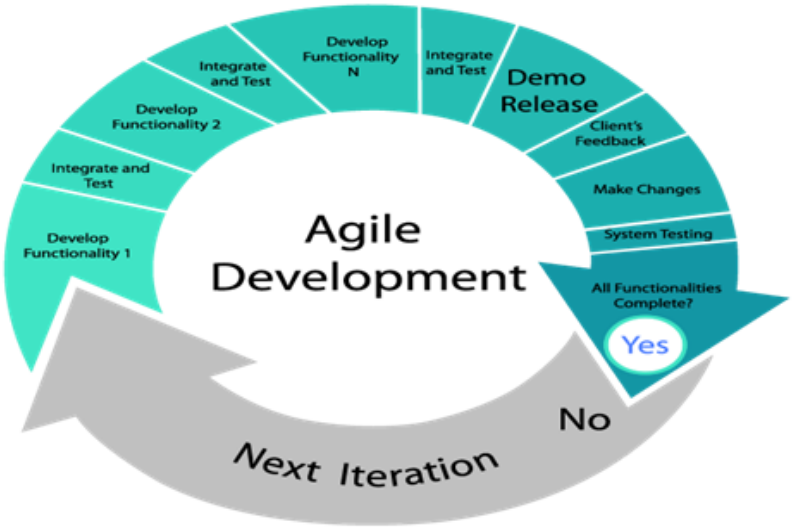


Figuur 2.2: RUP ontwikkel cyclus

2.2.3 Gebruik van Agile

Het ontwikkelen van software volgens de Agile methode gaat uit van een aantal principes. De eerste is individuen en interacties boven processen en tools. Het belangrijkste van het ontwikkelen vanuit het Agile perspectief is de tevredenheid van de klant door middel van snelle en doorlopende aflevering van waardevolle software. In vergelijking met RUP biedt Agile een aantal voordelen in dit gebied. De belangrijkste zijn het korter maken van de communicatielijnen en het betrokken maken van groepen bij het ontwikkelproces [20]. Dit gaat verder dan alleen het project voltooien en afleveren wat de klant in eerste instantie vroeg. Er vinden veel veranderingen in de eisen van het project plaats, door goed contact met de klant te houden kan ervoor gezorgd worden dat het product naar volle tevredenheid wordt afgeleverd. Een tweede principe is dat veranderingen in eisen en wensen door

een van de stakeholders van het project welkom zijn. Het maakt niet uit of dit aan het begin of aan het einde van het project is. In veel vakgebieden vinden veel veranderingen plaats, het is vaak niet mogelijk om aan het begin al te zien welke veranderingen plaats zullen vinden. Het accepteren van de veranderingen is makkelijker dan het proberen te voorkomen van veranderingen. Een derde principe is dat er elke paar weken of elke paar maanden een nieuwe versie van de software afgeleverd wordt. Het gaat er hier niet om dat het product dan naar de klant toe afgeleverd wordt, maar dat in ieder geval er een nieuwe interne versie is. Op deze manier kan iedereen leren van de obstakels die gevonden zijn in de laatste cyclus, om vervolgens weer verder te gaan (Figuur 2.3) [1]. Om te kunnen meten in welk stadium een project zich bevindt, wordt er in eerste instantie bepaald welk gedeelte van de software werkt. Dit wordt bewerkstelligd door het project op te splitsen in verschillende stappen. Op deze manier kan duidelijk en objectief bepaald worden in welk stadium een project zich bevindt. Het aandacht geven aan goed ontwerpen helpt bij de flexibiliteit. Een ontwerp maakt het mogelijk om veranderingen door te voeren op een wijze waarbij het snel duidelijk is welke gevolgen dit heeft bij de ontwikkeling van het project. Om deze flexibiliteit te kunnen bieden is het van belang dat de code van hetzelfde hoge niveau is als het ontwerp [11].



Figuur 2.3: Agile ontwikkel cyclus

2.3 Modellen

Modelleren is het ontwerpen van software applicaties voordat er begonnen wordt met het schrijven van de code. [3]. Dit is goed te vergelijken met het bouwen van een wolkenkrabber, ook hier begin je eerst met het maken van een model. Bij ontwikkelmethodes zoals Waterval, RUP en Agile, is een van de eerste stappen die doorlopen wordt, het maken van modellen. Dit is echter niet altijd het geval, het komt ook voor dat er pas in een later stadium modellen worden gemaakt. Er zijn verschillende modellen die gebruikt kunnen worden tijdens de ontwerpfase, elk met zijn voor en nadelen. Hier volgen een aantal modellen die mogelijk in de praktijk gebruikt worden bij het ontwikkelen van mobiele en/of webapplicaties.

2.3.1 Bedrijfslogica

Bedrijfslogica is over het algemeen een verzameling van formele en informele verklaringen over hoe er zaken worden gedaan [25]. Dit is logica die verband houdt met de bedrijfsvoering, als A gebeurt dan moet B gebeuren. Bijvoorbeeld als, bij een webwinkel een klant voor meer dan 50 euro besteld, dan is de verzending gratis. Door de groeiende afhankelijkheid van software, de groeiende complexiteit en de stijgende kosten van software, is het voor organisaties van steeds groter belang om de software te koppelen aan de bedrijfsvoering [6]. Het gebruik van bedrijfslogica is van groot belang bij het maken van modellen. Het uiteindelijke product moet aansluiten bij de bedrijfslogica, dus de modellen die gebruikt worden bij de ontwikkeling van het product ook [12].

2.3.2 Unified Modeling Language

De Unified Modeling Language(UML) is zoals door Conallen [9] beschreven de standaard taal waarin software applicaties beschreven worden, dit was in 1999. UML helpt de gebruiker bij het specificeren, visualiseren en documenteren van modellen van software systemen. Het is een manier om een abstract beeld te krijgen van de applicatie. Er zitten ook nadelen aan het gebruik van UML bij de ontwikkeling van webapplicaties. De standaard modelleer elementen sluiten niet voldoende aan bij het ontwikkelen van webapplicaties, vanuit de webapplicatie zouden de UML diagrammen er anders uitzien dan vanuit de native applicatie. Het onderzoek van Petre [23] sluit aan bij het onderzoek van Conallen, in het onderzoek van Petre is bij 50 bedrijven gekeken naar het gebruik van UML. Hieruit kwam naar voren dat in 35 van de onderzochte bedrijven er geen vorm van UML gebruikt wordt. In de overige bedrijven wordt een gedeelte van UML gebruikt en in sommige gevallen worden deze gedeeltes dan ook nog aangepast aan de eisen van het bedrijf. Er kwam wel naar voren dat er overal een vorm van formalisatie

gebruikt wordt. Dit laat zien dat de manier van modelleren sinds 1999 veranderd is, UML werd destijds vaak als niet praktisch gezien. Dit kan komen doordat het niet altijd goed aansluit bij de casus of omdat er extra uitleg bij een model nodig is om de context te kunnen begrijpen. Ondanks dat er nadelen aan het gebruik van UML zitten, zijn er ook duidelijke voordelen. Kulkarni [19] geeft aan dat het gebruik van UML er voor kan zorgen dat de code van het product minder complex wordt. Daarnaast ontstaat er een verhoogde productiviteit en meer uniformiteit in de code van het product, dit komt doordat de grote lijnen van het te maken product vast staan.

2.3.3 Use cases

Use cases kunnen gebruikt worden als het opstellen van requirements van een systeem kan niet voor voldoende duidelijkheid zorgt. Het voorbeeld dat Bittner [7] geeft maakt dit duidelijk: Bij het ontwerpen van een pinautomaat, is een requirement dat de klant geld moet kunnen opnemen. Deze requirement lijkt duidelijk, tot je er beter over na gaat denken. Want hoe moet dit dan precies gebeuren? Welke stappen moet de klant doorlopen? Zo zijn er nog meer vragen die beantwoord moeten worden om een goed systeem te kunnen ontwikkelen. Door gebruik te maken van use cases kan dit allemaal duidelijk worden. Een use case is een beschrijving van de stappen die door de gebruiker, dus in dit geval de klant, doorlopen moet worden om het systeem te gebruiken. Zo kan er voor elke gebruiker van het systeem een use case gemaakt worden.

2.3.4 User Stories

User stories zijn te vergelijken met use cases maar dan informeler. Ook de informatie die in een user story staat heeft niet enkel betrekking op interactie, het kan onder andere ook informatie over risico's van gebruikte technieken of beschrijvingen van unit tests bevatten. Bij user stories is het van belang dat het simpel wordt gehouden. Er wordt in enkele zinnen verteld wat de gebruiker wil doen [24].

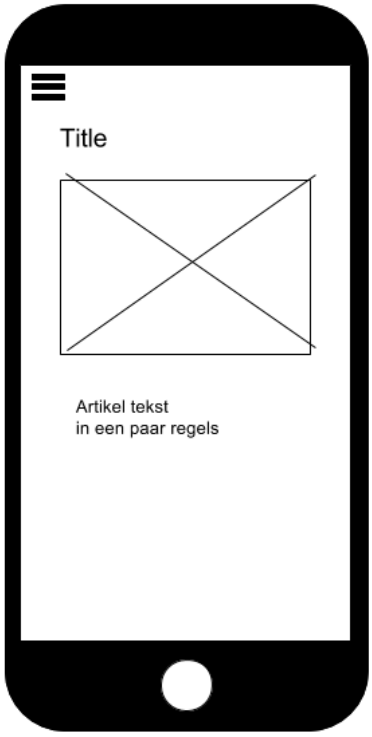
2.3.5 Proces Model

Als personen moeten samenwerken aan een gezamelijk project, dan is er een manier nodig om het werk te coördineren. Tijdens het coördineren kan er alvast rekening gehouden worden met problemen die zich mogelijk gaan voordoen. Bijvoorbeeld is er een kans dat als het project bijna klaar is dat er requirements eisen gaan veranderen. Het proces model zorgt er niet alleen voor dat er goed met problemen die waarschijnlijk zullen plaatsvinden. Het gaat over de workflow binnen de organisatie en in welke volgorde activiteiten plaatsvinden. Het zorgt er ook voor dat de organisatie kan leren van nieuwe

problemen, zodat deze in de toekomst beter opgelost kunnen worden. Het proces model is een kader waarmee software project specifieke processen gedefinieerd zijn. Dit model stelt de structuur, standaarden en relaties van verschillende proces elementen vast [15]. Bij de ontwikkeling van software worden proces modellen geïmplementeerd om zorgen met betrekking tot kosten, tijd, kwaliteit en dergelijke te beheren [18].

2.3.6 Wireframes

Een wireframe is een schets van de Grafische User Interface (GUI). Dit is een manier waarop er door middel van een schets een beeld gevormd kan worden over hoe het programma of de webpagina er uiteindelijk uit komt te zien (Figuur 2.4). Zo kun je een goed beeld krijgen van de plaatsing en grootte van verschillende onderdelen [22]. Wireframes of een andere manier van prototyping zijn goede manieren om te discussieren over verschillende scenario's en ideeën. Zo is het mogelijk dat cliënten en ontwikkelaars goed met elkaar kunnen communiceren [10].



Figuur 2.4: Wireframe voorbeeld

Hoofdstuk 3

Methode

Om te onderzoeken hoe het ontwikkelen van webapplicaties of mobiele applicaties in de praktijk gaat, wordt er in dit onderzoek gebruik gemaakt van een kwalitatieve onderzoekmethodiek en dan specifiek het gebruik van interviews om data te verzamelen. De keuze voor een kwalitatief onderzoek is gemaakt vanwege de flexibiliteit die dit biedt tijdens het onderzoek, aangezien niet van tevoren duidelijk is wat de mogelijke antwoorden op de vragen zijn.

3.1 Testpersonen

Er wordt een interview afgenomen bij zes personen die voor vijf verschillende organisaties werken. Ook zijn het bedrijven die verschillen in aantallen werknemers, om een zo breed mogelijk beeld te krijgen van de praktijk. De personen zijn allemaal Nederlandse mannen tussen de 25 en 50 jaar oud, die werken als software ontwikkelaar voor organisaties met een tiental werknemers tot aan organisaties met honderden werknemers.

3.2 Interview

In dit onderzoek wordt gebruik gemaakt van een semi gestructureerd interview, dit is een interview met vooropgestelde vragen waar van afgeweken kan worden. Op deze manier kan er dieper ingegaan worden op bepaalde vragen mocht dit nodig zijn, met als doel zoveel mogelijk informatie te verzamelen. Ook wordt door middel van een semi-gestructureerd interview ervoor gezorgd dat er geen onderdelen vergeten of overgeslagen worden. Een ander doel van het semi-gestructureerde interview is om het interview op een wetenschappelijke manier te laten plaatsvinden, maar om er tegelijkertijd voor te zorgen dat het interview toch het open karakter behoudt om op deze wijze zoveel mogelijk informatie te vergaren. Deze interviews worden opgenomen met een recorder en

vervolgens getranscribeerd. Het verder analyseren wordt gedaan aan de hand van deze transcripten. Het interview is op te delen in drie onderdelen (algemeen, eigen ervaring, overige), waarbij vragen opgesteld zijn die de basis vormen. Aan de hand van het antwoord van de geïnterviewde kan er door middel van extra vragen dieper op het onderwerp worden ingegaan.

Algemeen

Aan het begin van het interview wordt eerst gevraagd naar algemene informatie over de geinterviewde om op deze manier een correct beeld over de werkzaamheden en dagelijkse bezigheden te kunnen vormen. Dit wordt gedaan aan de hand van de volgende vragen:

- Zou u wat over uzelf kunnen vertellen?
- Wat doet u binnen de organisatie?

Eigen ervaring

Het tweede deel van het interview bestaat uit vragen over de werkwijze die gehanteerd wordt. Door de geïnterviewde het proces dat doorlopen wordt tijdens het ontwikkelen te laten vertellen, is het mogelijk een duidelijk beeld van de werkwijze te krijgen. Door eerst het complete proces door te lopen en vervolgens door te vragen op de onderdelen die interessant kunnen zijn voor dit onderzoek.

- Als er een nieuwe functionaliteit gemaakt moet worden, hoe gaat dat in zijn werk?
- Wordt er dan ook nog gebruik gemaakt van modellen of tekeningen?

Op het moment dat er geen duidelijk antwoord is, wordt er doorgevraagd door van de volgende modellen te vragen of deze gebruikt worden. Hierbij worden meerdere mogelijkheden tegelijkertijd genoemd om zo de invloed die het noemen van deze modellen heeft op de antwoorden van de geïnterviewde te minimaliseren.

- Proces modellen / flow charts
- Datamodellen
- bedrijfslogica
- Use Cases / User Stories
- Wireframes
- Functionaliteiten in termen van input/output

De laatste vraag van dit onderdeel gaan over de mening van de geïnterviewde.

- Is dit een manier van werken die u prettig vind of zou u graag aanpassingen zien?

Overige

Naast de ervaring bij de huidige werkgever is het ook mogelijk dat de geïnterviewde bij andere organisaties gewerkt heeft, ook is hier de mogelijkheid om andere vragen die tijdens het interview naar voren kwamen te stellen.

- Heeft u ervaring bij andere bedrijven, hoe daar gewerkt wordt?
- Ziet u in uw omgeving dezelfde werkwijze als bij dit bedrijf?

3.3 Analyse

Aan de hand van de transcripten zal de analyse plaatsvinden. Er zal met behulp van de in het theoretisch kader beschreven ontwikkelmethodes en modellen gekeken worden naar overeenkomsten met de praktijk. De analyse zal worden gedaan in drie stappen [5]. Eerst worden de transcripten gecodeerd. Het coderen van de transcripten houdt in dat er antwoord voor antwoord gekeken wordt naar de concepten die er genoemd worden. De codes die gecodeerd worden staan gedeeltelijk van te voren vast, dit zijn de concepten die in het theoretisch kader genoemd zijn. Als een concept nieuw is wordt deze toegevoegd aan de lijst met codes. Nadat de codes zijn geïdentificeerd worden deze samengevoegd in categorieën. Deze categorieën worden vervolgens gebruikt om na te gaan of er overeenkomsten zijn met concepten die in het theoretisch kader benoemd zijn.

Hoofdstuk 4

Resultaten

In dit hoofdstuk staan de resultaten van dit onderzoek. Deze resultaten zijn de categorieën die bestaan uit de coderingen van de transcripten. Vervolgens wordt er uitleg gegeven over de inhoud van deze categorieën. Aan de hand van deze categorieën wordt een vergelijking gemaakt met het theoretisch kader.

4.1 Codering

De verschillende interviews zijn na het transcriberen gecodeerd aan de hand van onderwerpen die uit het theoretisch kader naar voren zijn gekomen of er wordt een nieuwe code toegevoegd als deze niet in het theoretisch kader stond. Hieronder volgen twee voorbeelden van het coderen. Het is mogelijk dat een tekst meerdere codes krijgt, een code kan slechts eenmaal voorkomen in het antwoord op een vraag. Op het moment dat dezelfde code in het volgende antwoord van de geïnterviewde weer van toepassing is, wordt deze opnieuw dusdanig gecodeerd. Het aantal dat genoemd staat, is het aantal maal dat de desbetreffende code is gegeven aan een antwoord op een van de interviewvragen van alle afgenomen interviews.

Voorbeeld 1

Het volgende voorbeeld komt uit interview 1

Vooral kleine dingen, dat je denkt van. Weet je wel als het redelijk duidelijke use case is. Bijvoorbeeld als het iets is van de klant heeft een of andere slider op de website, dat is iets wat ik deze week even had gefixt. Om de zoveel seconden vervangt hij een plaatje en daar wouden we dan meldingen van Product A naadloos in verwerken. Dat is eigenlijk relatief simpel met wat het moet worden. Dus dan is de user interface ook relatief duidelijk dus dan overleg je even van zullen we het zo en zo gaan doen. Dus dan is het eigenlijk redelijk snel van top, laten we het gaan doen. Het is niet

zo dat daar van tevoren alles helemaal voor wordt dicht geplemt. Maar het is wel echt als het wat grotere dingen zijn, dat architecten daarover nadenken.

De eerste codering is als volgt gedaan: *Weet je wel als het redelijk duidelijke use case is. Bijvoorbeeld als het iets is van de klant heeft een of andere slider op de website, dat is iets wat ik deze week even had gefixt. Om de zoveel seconden vervangt hij een plaatje en daar wouden we dan meldingen van Product A naadloos in verwerken. Dat is eigenlijk relatief simpel met wat het moet worden. Dus dan is de user interface ook relatief duidelijk dus dan overleg je even van zullen we het zo en zo gaan doen.*

Er wordt aangegeven dat als de use case redelijk duidelijk is dan vindt er alleen even overleg plaats, hoe de oplossing gemaakt gaat worden. Dit zijn direct twee codes die genoemd worden. Ten eerste maken ze gebruik van use cases, namelijk dat er een slider op de website zit en dat meldingen hiervan in product A verwerkt moeten worden. Ten tweede wordt er aangegeven dat om tot een oplossing te komen er overleg plaatsvind wat precies te maken. Hier zijn vervolgens de codes 'use case' en 'overleg wat precies maken' aan gegeven.

De derde code zit in het laatste deel van dit antwoord: *Maar het is wel echt als het wat grotere dingen zijn, dat architecten daarover nadenken.*

Dit laat zien dat er bij ingewikkelde problemen er door een architect over een oplossing wordt nagedacht. De code bij deze zin is 'architect'.

Voorbeeld 2

Het volgende voorbeeld komt uit interview 3.

Nou dan hebben we duidelijk de oude stempel. Die zeggen we beginnen gewoon met maken en zien wel waar het strand. Dat is dus bij sommige subprojecten zo, er zijn een paar grote lijnen die uitgelegd zijn en die thematisch bepaalde problemen moeten oplossen. En enkele van die themas die komen terug in een sprint van een lange duratie. Dus we hebben die wekelijkse sprint, maar we hebben ook de grote generieke sprint, dus gewoon van onze huidige development versie. En daarin zitten vaak die issues die bij vlagen niet goed neergezet zijn. Die eigenlijk helemaal niet uitgedacht zijn. Een tekort aan informatie. Maar je hebt ook weer aan de andere kant gevallen dat het helemaal duidelijk is. Dat er eventueel zelfs een design bij zit. Implementatie specifieke dingen. Van te voren is het sinds de laatste drie maanden is het dat voor de sprint begint. Moet ontwikkelaars gevraagd worden, hoe lang ga je erover doen. Dus dan kan er een reele tijdsplanning inkomen. Op basis van dat kun je ook een reele periode afspreken dat de versie af moet komen. Met de gevraagde functionaliteiten.

De eerste codering van dit antwoord is als volgt gedaan: *Die zeggen we*

beginnen gewoon met maken en zien wel waar het strand. Dat is dus bij sommige subprojecten zo, er zijn een paar grote lijnen die uitgelegd zijn en die thematisch bepaalde problemen moeten oplossen

Er wordt in het begin aangegeven dat er direct begonnen wordt met ontwikkelen maar vervolgens wordt in meer detail uitgelegd dat er toch grote lijnen zijn. Dit betekent dus dat er niet helemaal zonder ontwerp gewerkt wordt, maar ook niet helemaal met behulp van een ontwerp. Dit is zodoende gecodeerd als 'soms een ontwerp'.

De tweede codering is als volgt gedaan: *En daarin zitten vaak die issues die bij vlagen niet goed neergezet zijn. Die eigenlijk helemaal niet uitgedacht zijn. Een tekort aan informatie.*

In deze zinnen wordt duidelijk verteld dat er bij de issues niet altijd goed vermeld staat wat er precies gedaan moet worden en dat hier niet goed is over nagedacht. Dit is dus gecodeerd als 'ongeveer een doel'.

Vervolg staat er de volgende zin: *Maar je hebt ook weer aan de andere kant gevallen dat het helemaal duidelijk is. Dat er eventueel zelfs een design bij zit*

Deze zin gaat over het feit dat er gevallen zijn waar er zelfs een ontwerp bij zit. Deze zin zou als 'soms een ontwerp' gecodeerd worden maar aangezien vier zinnen erboven hetzelfde staat wordt deze niet als dusdanig gecodeerd.

Code	aantal
ongeveer een doel	7
doelen veranderen	6
overleg over wat precies maken	6
requirements	4
minder volgens stramienen	4
schetsen	3
prototype	3
terugkoppeling	3
uml	3
gebrek aan controle	3
scrum	3
soms ontwerp	3
user stories	2
productie testen	2
use cases	2
interface ontwerp	2
documentatie	2
flow beschrijving	1
a b testen	1
architect	1
functioneel ontwerp	1
niet hele systeem bekend	1
niet hiërarchisch	1
waterval	1
minimalistisch werken	1
input/ output	1
informeel	1
prince2	1

4.2 Categorieën

Deze onderwerpen zijn samen te voegen tot verschillende categorieën. Sommige onderwerpen behoren tot meerdere categorieën. De categorieën zijn tot stand gekomen door te kijken naar onderwerpen die in meerdere interviews naar voren kwamen en tegelijkertijd bijdragen aan het

beantwoorden van de onderzoeksvraag van het huidige onderzoek.

Ontwikkel methodes	waterval
	prince2
	scrum
Direct beginnen	prototype
	soms ontwerp
	ongeveer een doel
	minder volgens stramienen
	minimalistisch werken
Eerst ontwerpen	overleg over wat precies maken
	requirements
	schetsen
	uml
	user stories
	soms ontwerp
	use cases
	interface ontwerp
	documentatie
	architect
	functioneel ontwerp
	input/ output
	terugkoppeling
Informeel werken	niet hiërarchisch
	gebrek aan controle
	minder volgens stramienen
	overleg over wat precies maken
	informeel
Ontwerp testen	a b testen
	productie testen
	prototype
Veranderingen	doelen veranderen
	ongeveer een doel
	overleg over wat precies maken
	terugkoppeling

Ontwikkelmethodes

Er worden drie verschillende ontwikkelmethodes genoemd. Deze ontwikkelmethodes zijn van belang bij het ontwikkelproces dat doorgemaakt wordt. Ook als de ontwikkelmethode niet precies volgens de officiële richtlijnen gevolgd wordt. Niet alle ontwerpmethodes werken goed samen met alle ontwikkelmethodes.

Het blijkt dat er twee manieren zijn waarop ontwikkelmethodes worden toegepast. Dit is of heel strak volgens de richtlijnen of minder strak volgens de richtlijnen. Dit lijkt samen te hangen met de grootte van de organisatie. Bij de grote organisaties (honderden werknemers) zijn de taken van de medewerkers duidelijk verdeeld en wordt er volgens richtlijnen gewerkt om het proces beheersbaar te houden. Zoals het volgende citaat duidelijk maakt: *We werken volgens scrum/agile en dat begint al bij de ontwerpfase waarbij er vrij vroeg schetsen gemaakt worden.* Dit is zoals het gebruik van agile in de theorie beschreven is. Bij de kleinere organisaties (Een tiental tot enkele tientallen werknemers) worden verschillende taken door dezelfde persoon uitgevoerd. Dit blijkt uit het volgende citaat: *Dus als ik grote wijzigen wil doen dan loop ik even bij hem langs, van hoe gaan we dat doen?.* De ontwikkelaar maakt de keuze dat er een grote verandering doorgevoerd moet worden en denkt mee aan een oplossing. Zo gebeurt het dat er begonnen wordt met ontwikkelen voordat de requirements of user stories helemaal duidelijk zijn. Hier worden de eerste stappen die in de theorie genoemd staan op een andere manier uitgevoerd. In plaats van een software architect, bedenkt de ontwikkelaar zelf een oplossing en overlegd over deze oplossing met andere ontwikkelaars.

Direct beginnen

Er komen verschillende manieren van werken naar voren waarbij onder andere direct begonnen wordt met het maken van het programma of de functionaliteit. Het direct beginnen met het ontwikkelen, is op te delen in twee onderdelen. Het eerste is om een prototype te maken, dit wordt gebruikt om te kijken of het idee dat er is, goed genoeg is om volledig te ontwikkelen.Het volgende citaat maakt dit duidelijk: *reisplanner gaat over een tijdje weer wat nieuws worden en we willen dan ook gaan A →B testen. Dus gaan kijken van we voegen nu een feature toe, werkt dat? Helpt dat mensen om beter hun ding te doen of werkt het misschien beter als we de knop daar zetten of geel maken of blauw maken.* Het tweede is daadwerkelijk beginnen met het ontwikkelen van het product. De redenen hiervoor hangen samen. Er is vaak geen ontwerp, omdat het doel niet helemaal duidelijk is en werknemers enige vrijheid willen hebben tijdens het werken. In sommige gevallen heeft dit ook met de leeftijd van het personeel te maken, zoals het volgende citaat duidelijk maakt: *Nou dan hebben we duidelijk de oude*

stempel. Die zeggen we beginnen gewoon met maken en zien wel waar het strand. Het direct beginnen met ontwikkelen komt niet overeen met de theorie, er is namelijk geen ontwerpfase.

Bedrijfslogica is een belangrijke basis als er direct begonnen wordt met het ontwikkelen. In deze gevallen heeft de ontwikkelaar het gevoel dat hij weet hoe de zaken gedaan worden en dit dus ook op een correcte wijze om kan zetten naar een programma. Doordat er snel resultaat geleverd moet worden, wordt er ook gebruik gemaakt van een prototype. Deze prototypes zijn of gedeeltelijk werkend of een representatie van het eindproduct en geven de opdrachtgever en/of gebruiker een idee van het uiteindelijke product. Het gaat hierbij dan ook in veel gevallen om de grafische weergave die te vergelijken is met een wireframe. Hierbij moet wel vermeld worden dat het product er net zoals het eindproduct uitziet en niet als een schets.

Eerst ontwerpen

Er zijn verschillende manieren waarop de ontwerpfase gebruikt wordt. Dit gaat van heel formeel tot heel informeel. Bij de organisaties waar het ontwerpen op een formele manier gebeurt, is er een software architect. Het volgende citaat maakt dit duidelijk: *Eerst worden er requirements geschreven. Op basis daarvan worden use cases geschreven. Op basis daarvan gaan de systeemarchitecten en functioneel ontwerpers, die gaan functioneel ontwerpen maken.* Als de ontwerpfase niet op deze wijze plaatsvindt bij een organisatie, dan wordt er vaak op een informele wijze toch geprobeerd om duidelijk te krijgen wat de eisen van product zijn. Dit gebeurt dan voornamelijk door te overleggen met collega's of door in de issue tracker een oplossing te beschrijven en daar vervolgens feedback op te krijgen. Het volgende citaat maakt dit duidelijk: *Het is meestal in JIRA een issue waar iemand gewoon iemand een ontwerpje neer zet. We gaan het zo en zo doen. Het werkt zo en zo. Dit praat daar mee en dit praat daar mee. Dan gaan andere mensen ernaar kijken. Die laten dan in JIRA comments achter.*

Bij de grote organisaties worden er veel verschillende soorten modellen gebruikt. Bij deze organisaties wordt ook strak volgens de Agile ontwikkelmethode ontwikkeld. Dus er worden user stories gemaakt, hier worden use cases van gemaakt. Er wordt een functioneel ontwerp gemaakt. Deze modellen en het gebruik ervan komen overeen met de theorie. Bij de kleinere organisaties is dit allemaal wat minder transparant. De ontwikkelaar heeft vaak zelf de vrijheid om te bedenken hoe hij iets wil gaan maken en in sommige gevallen zelfs wat. In bepaalde mate worden er ook requirements opgesteld, dit gebeurt dan met behulp van de issuetracker, dit is meteen de plaats waar collega's feedback leveren op een voorgestelde oplossing. Aangezien de modellen die gebruikt worden bij de kleinere organisaties niet helemaal vast staan, is de overeenkomst met de theorie niet

volledig. De ontwikkelaar bepaald zelf welke modellen er gebruikt worden. Zo worden er tekeningen gemaakt die lijken op UML.

Informeel werken

Bij de kleinere organisaties wordt aangegeven dat het werk niet hiërarchisch gebeurt. Hiermee wordt bedoeld dat het ontwikkelproces niet helemaal stap voor stap doorlopen wordt. Dus dat er niet eerst een ontwerper aan de slag gaat, vervolgens een ontwikkelaar, enzovoorts. De ontwikkelmethode komt dan in grote lijnen overeen met de theorie, echter worden niet altijd alle stappen volledig doorlopen. Er worden zowel voor- als nadelen van deze manier van werken genoemd. Voordelen die genoemd worden zijn dat er makkelijk overlegd kan worden als er een probleem is. Je loopt gewoon even bij je collega langs en bedenkt een oplossing van het huidige probleem. Zoals het volgende citaat duidelijk maakt: *Hoe kleiner het is, hoe meer het op hoofdlijnen is van dit moet het ongeveer doen. Hoe kleiner het ook is. Hoe eerder je ook even stuurt van hoe vind je dat dit eruit ziet.* Een ander belangrijk pluspunt is dat deze manier van werken, het werk creatief houdt. Je moet nadenken over een probleem en een oplossing bedenken, niet gewoon een ontwerp omzetten in code. Een groot nadeel van het informeel werken is, dat er een gebrek aan controle is. Het is mogelijk dat slechte code zijn weg weet te vinden in het uiteindelijke product.

Het spiraal model dat bij RUP gebruikt wordt, komt naar voren bij het informeel werken. Er wordt in veel gevallen door een ontwikkelaar begonnen aan een nieuwe functionaliteit. Dit begint klein en simpel en wordt uitgewerkt richting een volledig werkende functionaliteit. Op het moment dat de ontwikkelaar problemen tegenkomt dan worden deze opgelost door met collega's te overleggen. Zo groeit het aantal personen dat zich bezig houd met deze functionaliteit en gaat de groeit de cyclus van de ontwikkeling om zo een spiraal te vormen.

Ontwerpen testen

Het testen van software wordt ook op andere manieren gedaan. In sommige gevallen wordt er een prototype gemaakt om hiermee te testen of het idee dat er is, ook daadwerkelijk bruikbaar is. Verder komt het ook voor dat een programma/functionaliteit wordt toegevoegd of gewijzigd en dat er vervolgens in productie gekeken wordt of dat het werkt. Hierbij gaat het dan zowel om het feit dat er afgewacht wordt of er bugs in de software zitten als dat het ontwerp ook door de gebruiker als prettig ervaren wordt. Zoals het voorbeeld uit het volgende citaat: *Helpt dat mensen om beter hun ding te doen of werkt het misschien beter als we de knop daar zetten of geel maken of blauw maken.* Op deze manier wordt er gekeken of de verandering een goede keuze was. Het unittesten gebeurt alleen bij de grote organisaties,

bij de kleinere organisaties wordt dit vaak niet gedaan vanwege het niet aanwezig zijn van een tester. Aangezien deze werkwijze standaard is, past dit binnen het proces model dat in de theorie genoemd is. In dit model wordt de workflow beschreven zonder dat de inhoud van deze workflow vaststaat.

Veranderingen

Het blijkt dat er veel veranderingen plaatsvinden tijdens het ontwikkelen van de software. Het komt bij een aantal bedrijven voor dat deze meteen doorgevoerd moeten worden. Dit moet direct meegenomen worden in de huidige iteratie van het ontwikkelproces, dit zorgt ervoor dat de doelen vaak niet gehaald worden. Zoals het volgende citaat duidelijk maakt: *Wat er bij ons nog weleens gebeurt is dat er of de sprint langer doorloopt of dat er issues bij ingeschoten worden.* Er zijn verschillende bronnen voor deze veranderingen. Dit kan zijn doordat het doel veranderd, het doel niet helemaal duidelijk was of dat er terugkoppeling komt op hetgeen dat al ontwikkeld is. Het veranderen van doelen, is iets dat in de theorie bij zowel het Rational Unified Proces als bij de Agile ontwikkelmethode beschreven. Het is wel van belang dat de veranderingen op bepaalde momenten doorgevoerd worden. Dit gebeurt in de praktijk dus ook op andere momenten, er kan daarom minder goed met deze verandering worden omgegaan.

Hoofdstuk 5

Conclusies

In dit onderzoek is onderzocht of er concepten uit de digitale architectuur gebruikt worden bij het ontwikkelen van web of mobiele applicaties. Uit dit onderzoek komt naar voren dat het aantal werknemers van de organisatie grote invloed heeft op het gebruik van concepten uit de architectuur. Bij de grote organisaties(honderden werknemers) gaat alles volgens een vast patroon. Dit gebeurt omdat het proces anders niet te beheersen is. Deze organisaties maken gebruik van de agile/scrum ontwikkelmethode en gebruiken deze zoals in de theorie beschreven. Er wordt dus eerst een architectuur opgesteld voordat er begonnen wordt met ontwikkeling.

Bij de kleinere organisaties (een tiental tot enkele tientallen werknemers) ligt dit anders. Deze organisaties willen ook graag volgens de agile/scrum ontwikkelmethode werken, dit blijkt in de praktijk toch net iets anders te liggen. Of dit komt doordat werknemers vastzitten in hun werkpatroon of het feit dat er minder personeel beschikbaar is om alle taken uit te voeren, is nog de vraag. Er komt naar voren dat het nut van een goede architectuur duidelijk is, echter de tijd om deze ook te maken is er vaak niet. Er wordt bij deze organisaties voor een meer informele manier van werken gekozen. Iedereen zorgt goed voor zijn onderdeel en als er een probleem is dan wordt dit onderling overlegd om zo tot een goede oplossing te komen.

De geïnterviewden geven aan dat ze voor zichzelf op papier zetten wat ze willen gaan maken. Een uitdaging hierin is dat de architect dan ook de ontwikkelaar is (en vaak ook de domein expert). Doordat er meerdere functies in een persoon verenigd zitten vergroot dit de kans op fouten, doordat er een tunnelvisie ontstaat bij de ontwikkelaar. Dit wordt in de praktijk vaak opgelost door even met een collega te overleggen, op deze manier denkt ook iemand anders over het probleem na. De reden dat een werknemer zowel architect als ontwikkelaar is lijkt te liggen in de grootte van de organisatie, er is simpelweg niet genoeg personeel om het op te splitsen. Een van de oplossingen die vaak gebruikt wordt, is dat er een prototype ontwikkeld wordt. Op deze manier wordt geprobeerd of

iets werkt en als dit niet zo is dan wordt het weer aangepast. Er kan ook een voordeel zitten aan het feit dat de ontwikkelaar de architectuur maakt. De ontwikkelaar weet namelijk precies wat er bedoeld wordt bij de architectuur, er kunnen geen fouten ontstaan door interpretatie van het werk van een ander. Ontwikkelaars maken regelmatig tekeningen voor zichzelf om duidelijk te krijgen wat er ontwikkeld moet worden, deze tekeningen lijken op UML diagrammen. Een grote uitdaging bij de kleine organisaties is dat het niet altijd duidelijk is wat het doel precies is of dat als het doel duidelijk is, deze gedurende het ontwikkelen verandert. Dit maakt het erg lastig om goede requirements of use cases op te stellen. Er worden dus wel concepten uit de architectuur gebruikt, maar het is summier.

Al met al, is er in dit onderzoek gevonden dat de grotere organisaties gebruik maken van de Agile ontwikkelmethode en elke stap hiervan volledig doorlopen, bij de kleinere organisaties is er niet altijd een duidelijke ontwerpfase van het product. Een optie voor vervolgonderzoek is dat er gezocht kan worden naar een methode om architectuur op een snelle en directe manier te kunnen gebruiken bij het ontwikkelen van mobiele/webapplicaties.

Bibliografie

[1] http://www.matrix-soft.org/agile-development-methodology/.

[2] http://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.

[3] http://www.uml.org.

[4] http://www.volkskrant.nl/binnenland/slechte-start-a73-11-afsluitingen-in-10-dagen a888087/.

[5] Soon K Bang, Sam Chung, Young Choh, and Marc Dupuis. A grounded theory analysis of modern web applications: knowledge, skills, and abilities for devops. In *Proceedings of the 2nd annual conference on Research in information technology*, pages 61–62. ACM, 2013.

[6] Victor R Basili, Jens Heidrich, Mikael Lindvall, Jürgen Münch, Myrna Regardie, Dieter Rombach, Carolyn Seaman, and Adam Trendowicz. Linking software development and business strategy through measurement. *arXiv preprint arXiv:1311.6224*, 2013.

[7] Kurt Bittner. *Use case modeling*. Addison-Wesley Longman Publishing Co., Inc., 2002.

[8] Andre Charland and Brian Leroux. Mobile application development: web vs. native. *Communications of the ACM*, 54(5):49–53, 2011.

[9] Jim Conallen. Modeling web application architectures with uml. *Communications of the ACM*, 42(10):63–70, 1999.

[10] Mogamat Razeen Davids, Usuf ME Chikte, and Mitchell L Halperin. Development and evaluation of a multimedia e-learning resource for electrolyte and acid-base disorders. *Advances in Physiology Education*, 35(3):295–306, 2011.

[11] Martin Fowler and Jim Highsmith. The agile manifesto. *Software Development*, 9(8):28–35, 2001.

[12] Alfonso Fuggetta and Elisabetta Di Nitto. Software process. In *Proceedings of the on Future of Software Engineering*, pages 1–12. ACM, 2014.

[13] Carmine Giardino, Nicolo Paternoster, Michael Unterkalmsteiner, Tony Gorschek, and Pekka Abrahamsson. Software development in startup companies: The greenfield startup model. 2013.

[14] Laura Hokkanen, Kati Kuusinen, and Kaisa Väänänen. Early product design in startups: Towards a ux strategy. In *Product-Focused Software Process Improvement*, pages 217–224. Springer, 2015.

[15] Watts S Humphrey and Marc I Kellner. Software process modeling: principles of entity process models. In *Proceedings of the 11th international conference on Software engineering*, pages 331–342. ACM, 1989.

[16] Ming Huo, June Verner, Liming Zhu, and Muhammad Ali Babar. Software quality and agile methods. In *Computer Software and Applications Conference, 2004. COMPSAC 2004. Proceedings of the 28th Annual International*, pages 520–525. IEEE, 2004.

[17] Mona Erfani Joorabchi, Ali Mesbah, and Philippe Kruchten. Real challenges in mobile app development. In *Empirical Software Engineering and Measurement, 2013 ACM/IEEE International Symposium on*, pages 15–24. IEEE, 2013.

[18] Rupinder Kaur and Jyotsna Sengupta. Software process models and analysis on failure of software development projects. *arXiv preprint arXiv:1306.1068*, 2013.

[19] Vinay Kulkarni. Model driven software development. In *Modelling Foundations and Applications*, pages 220–235. Springer, 2013.

[20] Ramon Noordeloos, Christina Manteli, and Hans Van Vliet. From rup to scrum in global software development: A case study. In *Global Software Engineering (ICGSE), 2012 IEEE Seventh International Conference on*, pages 31–40. IEEE, 2012.

[21] Jorge A Osorio, Michel RV Chaudron, and Werner Heijstek. Moving from waterfall to iterative development: An empirical evaluation of advantages, disadvantages and risks of rup. In *Software Engineering and Advanced Applications (SEAA), 2011 37th EUROMICRO Conference on*, pages 453–460. IEEE, 2011.

[22] Jeff Patton. Hitting the target: adding interaction design to agile software development. In *OOPSLA 2002 Practitioners Reports*, pages 1–ff. ACM, 2002.

[23] Marian Petre. Uml in practice. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 722–731. IEEE Press, 2013.

[24] Michael J Rees. A feasible user story tool for agile software development? In *Software Engineering Conference, 2002. Ninth Asia-Pacific*, pages 22–30. IEEE, 2002.

[25] Minhong Wang and Huaiqing Wang. From process logic to business logic—a cognitive approach to business process management. *Information & Management*, 43(2):179–193, 2006.

[26] Mina Zaminkar and Mohammad R Reshadinezhad. A comparison between two software engineering processes, rup and waterfall models. In *International Journal of Engineering Research and Technology*, volume 2. ESRSA Publications, 2013.

Appendix A

Interview 1

Interviewer wordt weergegeven als P.
Interviewde wordt weergegeven als I.
P: Kun je wat over jezelf vertellen? Wat je doet? Waar je werkt?
I: Yes, ja ik werk bij bedrijf A als costumer succes engineer. Dus dan maken we eigenlijk plugins voor product A. Dat is eigenlijk javascript wat voor een deel serverside met Nodejs draait en voor een deel eigenlijk het grootste deel clientside draait.
P: Je zegt dat je een Costumer succes engineer bent, wat is dat?
I: Eigenlijk zegt het heel weinig, je kan ook software engineer opschrijven. Het komt erop neer, we hebben verschillende teams en je hebt dan het platform team. Dat werkt echt met Java. Dat bouwt het platform en rest API's enzo. Dan heb je het plugin team, dat bouwt op basis van die rest API's, bouwen ze plugins voor connecties met andere systemen en data visualisaties. En eigenlijk allerlei dingen om daadwerkelijk business waarde toe te voegen. Dan heb je nog het constumer succes team, die zijn heel concreet bezig met problemen die klanten die op dat moment hebben. We zitten dan ook niet gebonden aan release cycles, we kunnen releasen wanneer we willen. We lossen eigenlijk problemen op die klanten op dat moment hebben, om meerwaarde voor de klanten te bieden.
P: En als jullie dan zo'n probleem tegenkomen of er moet een nieuwe functionaliteit ontwikkeld worden. Hoe gaat dat dan in zijn werk?
I: Eerst even een discussie over hoe we het gaan oplossen. Wat is nou de mooiste manier. We hebben daar wel verschillende plugin types. Die moet je eigenlijk zien als interfaces, die op verschillende manieren wordt aangeroepen. Je hebt bijvoorbeeld een listener om gegevens op een of andere manier af te luisteren en op te slaan. Dan heb je connecties, die is bedoeld om gegeven op te halen en weer om zich heen te pushen. Dus dan moet je een beetje kijken, hoe lossen we dit probleem op. Met wat voor voor type en hoe zetten we dat dan een beetje handig op, dat het ook schaalt enzo. Dat soort dingen eigenlijk. Meestal hebben klanten ook iets vergelijkbaars.

Dan moet je kijken wat is op dit moment de beste manier om het te doen. Heel veel systemen kun je tegenwoordig ook serverside koppelen. Maar dan is het al snel een batch koppeling, maar je hebt tegenwoordig ook een real time koppeling. Dan is het een javascript tag, dat javascript een async call heeft.
P: Als jullie er dan over nadenken hoe het probleem het beste opgelost kan worden. Hoe doen jullie dat nadenken? Met schetsen, Met bepaalde modellen.
I: Het is meestal in JIRA een issue waar iemand gewoon iemand een ontwerpje neer zet. We gaan het zo en zo doen. Het werkt zo en zo. Dit praat daar mee en dit praat daar mee. Dan gaan andere mensen ernaar kijken. Die laten dan in JIRA comments achter, we gebruiken JIRA om het proces te lijden. Anders krijg je weer van die meetings waar iedereen bij elkaar in een kamer zit. Het zijn toch vaak wel dingen waar je even over wilt nadenken. Op basis van die feedback wordt het ontwerp dan aangepast. Het is ook wel een beetje afhankelijk van hoe groot het ding is. Kijk als het 5 regels of 50 regels is, dan hoeft het allemaal niet. Maar meestal als het wat grotere dingen zijn dan, dan is het gewoon praktisch om er van te voren even over na te denken.
P: Houden jullie dan op die manier ook bezig als er bijvoorbeeld in de frontend iets moet wijzigen met hoe het eruit ziet? Of maak je dan een schets van hoe iets moet veranderen?
I:Ja als het echt user interface is van product A dan hebben we daar een designer voor. Die maakt in sketch ofzo een tool voor OSX. Daar kun je heel makkelijk user interfaces mee tekenen. Dus als ik grote wijzigen wil doen dan loop ik even bij hem langs, van hoe gaan we dat doen?
P: Dat lijkt een beetje op wirediagram?
I: Hij heeft eigenlijk gewoon alle user interface onderdelen van product A heeft hij daar ergens klaar staan. Dus dat kan hij een beetje in elkaar copy pasten. Ik probeer dat een beetje een op een over te maken in css. Als het een paar pixels afwijkt dan boeit dat niet natuurlijk niet. Maar hij maakt van die plaatjes en die plemt hij in JIRA van zo moet het worden, ongeveer.
P: Dus dat is eigenlijk gewoon een screenshot van de website hebt?
I: Praktisch is het een screenshot. Het is wel een beetje. Ook hier natuurlijk, als het iets heel kleins is, dat ik dan bij hem langs loop van ik was voor plan om het zo en zo te doen. Wat denk jij en even snel van laten we het zo en zo doen. Dan gaat hij er geen screenshots van maken. Daar zoeken we trouwens ook nog iemand voor. Dus als je nog iemand weet. Maar vooral ook bij grote dringen. Dan gaat hij wel echt eerst screenshots maken. Hoe dat proces van hem echt loopt. Volgens mij moet hij het dan ook echt eerst met allemaal andere mensen weer overleggen enzo. Als het echt iets groots is, dan gaat er van hem ook best veel werk in zitten.
P: Maar eigenlijk hebben jullie dus best veel vrijheid in hoe jullie dingen aanpakken? Jullie moeten wel met mensen overleggen maar als het iets

kleins is dan.
I: Vooral kleine dingen, dat je denkt van. Weet je wel als het redelijk duidelijke use case is. Bijvoorbeeld als het iets is van de klant heeft een of andere slider op de website, dat is iets wat ik deze week even had gefixt. Om de zoveel seconden vervangt hij een plaatje en daar wouden we dan meldingen van product A naadloos in verwerken. Dat is eigenlijk relatief simpel met wat het moet worden. Dus dan is de user interface ook relatief duidelijk dus dan overleg je even van zullen we het zo en zo gaan doen. Dus dan is het eigenlijk redelijk snel van top, laten we het gaan doen. Het is niet zo dat daar van tevoren alles helemaal voor wordt dicht geplemt. Maar het is wel echt als het wat grotere dingen zijn, dat architecten daarover nadenken.
P: Jullie maken dan ook use cases van hoe het dan moet gaan werken? Of krijg je specifiek wat je dan moet gaan maken? Van de architecten, je krijgt deze functionaliteit die dit doet, Of hoe ziet dat eruit wat je van de architect krijgt.
I: Dat is meer ook in JIRA. Dat is wel op hoofdlijnen, het is niet zo dat deze functie moet dit gaan doen. Maar wel we hebben deze listener en deze gegevens komen daarin en deze gegevens komen daaruit. Dus het is meer op data niveau dan op implementatie niveau. Het is meer een interface omschrijving dan. Het is wel als je kijkt dat bij dingen waar je met meerdere mensen aan werkt, dat het importeren en exporteren van klantgegevens. Dan heb je bijvoorbeeld een connector voor google drive. Daar ben ik dan nu mee bezig. We hebben er een voor s3 enzo. Daar hebben we een interface voor. Deze functies moeten in je javascript file. Dan plemp je die hier zo in en dan werkt het zeg maar automatisch. Dus ja voor dingen waar je met meerdere mensen aan werkt die dus ook een grotere impact hebben. Vooral iets wat voor meerdere klanten is hebben iets meer interfaces en wel iets meer beschrijving en meer regie ook, wat voor functies je moet implementeren.
P: Dus hoe ingewikkelder het wordt, hoe meer er overlegt wordt, hoe meer er gedocumenteerd wordt?
I: Ja ik denk, dat waar elk klein wissewasje een ander ontwerp gaat maken, dat is misschien ook een beetje overdreven. Hoe kleiner het is, hoe meer het op hoofdlijnen is van dit moet het ongeveer doen. Hoe kleiner het ook is. Hoe eerder je ook even stuurt van hoe vind je dat dit eruit ziet. Dan gaan ze daarop testen en als daar iets uit komt dan hoor je het wel.
P: Wat vind je van deze manier van werken?
I: Ik vind het altijd wel relaxed als ik zelf ook nog wat mag, kan bedenken. Als het helemaal uit gekristalliseerde is, dan is de creativiteit er ook een beetje uit. Dan is voor mij de lol er ook wel enigszins af. Het is ook wel, dat als je van die compleet in elkaar gehackte dingen, dan heb je ook wel eens dat je dan in andermans code zit. Dan zit het wel compleet zonder structuur is opgezet. Dan denk je ook wel van ja. Maar dat is denk ik ook wel een beetje afhankelijk van wat voor soort mensen en wat voor achtergrond ze hebben.

P: Wordt er ook nog naar gekeken van hoe mensen ze dingen, hoe mensen programeren?
I: Officieel hebben we code review. Nja ik heb er nog nooit iets van gehoort. Volgens mij gebeurt het niet zo heel veel. Ik weet wel dat bij ons de junior programmeurs, dat daar wel echt meer naar wordt gekeken. Dit komt ook ergens online te staan?
P: Het interview zelf niet.
I: Ok, Maar over het algemeen wordt er niet heel veel, gereviewd.
P: Ok, als je een beetje om je heen kijkt dan, bij andere organisaties. Zie je daar een zelfde manier van werken.
I: Nou bij de bedrijf B was het zo dat, daar had je dan in JIRA, als je dan iets af hebt klik je het door naar de review stap. Dan moest iemand anders erop aftekenen, van ik heb het gereviewed en het was ok of pas dit en dit nog even aan. Daarna klik je het door naar test. Dan gaat iemand anders het testen. Dat is hier natuurlijk niet. Ik denk ook wel dat dingetjes als misschien, Scrum en misschien algemeen ook proces. Dat daar iets te weinig van is. Ik snap ook wel dat, je bent een kleine organisatie en dan is het misschien ook niet zo nodig. Maar ik denk wel dat qua kwaliteit van software zouden we nog wel een stap kunnen maken.
P: Heeft het dan toch wel iets weg van het gebruik van Scrum? Omdat je niet echt deadlines hebt?
I: Oh, kijk we hebben wel stand-ups en sprint enzo. Dus het is echt niet zo dat we niets aan scrum doen. Er is ook een retrospective enzo. Maarja ik denk wel dat dingetjes zoals een strak review proces. Dat is ook misschien niet altijd even nodig hoor. Maar ik denk wel opzicht dat het wel chill zou kunnen zijn. Het is nu ook bijvoorbeeld het software testen. Doe je ook net iets te weinig aan. We hebben geen dedicated tester.
P: Dan wil ik je graag bedanken.

Appendix B

Interview 2

Interviewer wordt weergegeven als P.
Interviewde wordt weergegeven als I.
P: Kun je wat over jezelf vertellen? Wat je doet binnen de organisatie?
I: Mijn naam is I. Ik ben inmiddels vier jaar aan het werk als software developer. Ik ben begonnen bij Bedrijf A, dat is een detacheerder in Eindhoven. Vooral in embedded technologie. Voor hun heb ik bij verschillende klanten gezeten, zoals Bedrijf B in Groenloo. Daarna ben ik door gegaan naar een ander bedrijf. Bedrijf C dat is een detacheerder die zit in Den Bosch. Daar ben ik heen gegaan omdat ik meer groepsgebaseert wilde werken. Daar heb ik ook anderhalf jaar ofzo gezeten. Nu zit ik ongeveer een jaar bij bedrijf D, omdat ik liever bij een wat kleiner bedrijf wilde werken. Omdat ik daar wel benieuwd naar was. Ik had het idee dat, dat beter zou zijn. Dat dat minder organisatorische rompslomp zou zijn.
P: Dat kan, Wat doe je hier binnen bedrijf D?
I: Binnen bedrijf D ben ik software developer en security officer. Security officer klinkt heel zwaar, maar dat valt eigenlijk wel mee. Ik moet een beetje.. We scannen automatisch de beveiliging van de applicatie, dat laten we automatisch doen. Dat moet ik in de gaten houden, of daar nog treats uitkomen. Verder doe ik ook wat de rest ook doet. Zeg maar las ontwikkelaar. Er komen issues op de backlog en in de sprint. Die worden opgepakt en die rond ik dan af.
P: En dat is dan Mobiele of web of backend?
I: Backend voornamelijk, mobiele dat doet Michiel voornamelijk.
P: Als jullie dan de opdracht krijgen om een nieuwe functionaliteit te maken hoe gaat dat dan in zijn werk?
I: Voorheen was het zo dat er een hele globale omschrijving was van maak iets zodat we kunnen smsen. Nu wordt er meer met de ontwikkelaar samen gekeken, van wat moet er eigenlijk gebeuren. Want ik heb laatst iets gemaakt zodat er tijdzones aan afspraken kunnen worden gekoppeld en daar is ook met consultancy, dus mensen die met de klant communiceren. Die weten

wat er speelt, ook gecommuniceert. Zo is er meer betrokkenheid om samen tot een goede oplossing te komen.
P: Bij die eerste stap om samen tot een oplossing te komen. Maken jullie gebruik van, tekenen jullie nog dingen uit of..?
I: Er is niet echt een vast stramien voor denk ik. We kijken gewoon wat we zelf denken dat de beste oplossing is. Dat is natuurlijk lang niet. Je hoeft natuurlijk lang niet altijd UML diagram te maken of activity diagram te maken of iets. Niet alles hoeft helemaal vast te liggen.
P: Ik bedoel het ook niet dat het heel strak, dat je perce alles volgens die regels neerzet. Maar of je zelf bepaalde type tekeningen maakt of schetsen maakt. Al is het een schets ergens van. Of hoe je iets aan elkaar wilt koppelen.
I: Dat ligt heel erg aan de complexiteit van het issue. Ik vind zelf dat smsen is best een complex issue want je kunt er best heel veel kanten mee op. Wat doe je bijvoorbeeld als het nummer niet werkt..
P: Hoe deed je dat dan bij dat smsen? I: Omdat je tijdens het ontwikkelen loop je tegen de problemen aan. Dit komt omdat je van te voren niet goed genoeg hebt nagedacht. Als je van tevoren met de consultancy nadenkt over waar kunnen de klanten tegen aanlopen. Dan heb je al een beter beeld. Als je dat voor het ontwikkelen doet, heb je al een keer nagedacht. Dus dan kun je.
P: En als je die problemen dan tegenkomt, ga je dan terug naar de stap van laat er nog eens een keer over gaan nadenken ? Over hoe het in mekaar zit of blijf je toch snel doorgaan waarmee je bezig was.
I:Af en toe kun je denk ik die stap niet helemaal terug naar af maken. Dus dan moet je maar het beste ervan maken.
P: Dan zie je later wel van hoe het er uitkomt. Dan zie je later wel van of er nog doorontwikkeld moet worden, waar je mee bezig was.
I:Dus je vraag is van je hebt een probleem, je hebt er niet goed over nagedacht hoe je het gaat aanpakken. Je gaat toch beginnen met het probleem en komt er halverwege achter dat het probleem. Dat je misschien niet goede oplossing hebt gekozen. Dat je dan niet meer terug kunt. Wat is dan je vraag?
P: Hoe ga je dan toch verder? Ga je dan toch nog eens nadenken of schetsen maken om te kijken hoe je er wel uit kunt komen. Of blijf je het puur in je hoofd houden waar je mee bezig bent.
I: Ik denk dat als het complex is dat je altijd iets moet gaan tekenen of opschrijven of met andere mensen moet gaan bespreken. We hebben hier ook een grote kantoortuin, dus het is ook redelijk makkelijk om iemand anders even te vragen van hoe gaan we dat nou aanpakken.
P: Dus als het complex is, dan wordt er wel snel overlegt?
I:Ja er wordt wel snel overlegt.
P: Ok en maar bij dat ontwikkelen maken jullie gebruik van sprints of meer de waterfall manier van ontwikkelen?

I: We proberen wel met sprints te werken, maar wat bij sprints eigenlijk heel belangrijk is, is dat er een vaste tijdslimiet aan zit. En dat je een vast aantal issues hebt. Wat er bij ons nog weleens gebeurt is dat er of de sprint langer doorloopt of dat er issues bij ingeschoten worden. Die dan toch belangrijk zijn. Daar zijn we mee bezig om dat te verbeteren, maar dit blijft toch gebeuren. Wat we ook doen. We hebben sprints van de grote development sprints duren 4 weken en we hebben elke week een dag de patchsprint en daar kunnen de urgente dingen worden opgelost. Dat is om te voorkomen dat we eigenlijk continu dingen worden ingeschoten.
P: Dus jullie proberen met een andere manier van werken, dat die grote sprint. Die jullie eigenlijk willen gaan gebruiken ook kunnen gebruiken?
I: Dat daar eigenlijk geen issues bijkomen en dat die niet langer duurt. Maar dat lukt nog niet helemaal.
P:Met het ontwikkelen heb je dan ook nog te maken met business logic of is het allemaal gewoon puur een functionaliteit bouwen en dat je niet zoveel last hebt van wat bedrijven precies willen?
I:Ik denk dat alle functionaliteiten business logic zijn toch.
P: Ik bedoel van als je begint met het maken van een functionaliteit of je dan puurt met die functionaliteit bezig bent of dat je dan ook gebruik van maakt van de business logic van het bedrijf. Om te zorgen dat het ook gaat doen wat het moet doen. Of is die business logic niet zo van belang is?
I: Wat is de business logic dan volgens jou?
P: Jullie maken die enquete software. Dat het puur is dat de enquete software blijft werken of dat de functionaliteiten zo zijn dat ze niet afhankelijk zijn van hoe het systeem in elkaar zit.
I: Dat is ons systeem dat heeft niets met hun business te maken. Wat ik laatst wel had. Je kunt outlook invites zelf opbouwen in je code en die kun je dan ook versturen. Dan werkte die bij ons wel maar dan hebben we een klant en die heeft op zijn exchange server. Dat waar die inmiddels ontvangen worden. Daar gaat iets mis. Dat is dan hun business logic. Het gaat helemaal mis aan hun kant maar toch bij ons. Want wij moeten dan maar zorgen dat ons formaat bij hun past.
P: Ja dat bedoel ik, dat jullie met de klant.
I: Ja dat zijn lastige issues. Want je hebt af en toe het inzicht niet. Ik kan niet op hun exchange server. Hun contact persoon weet eigenlijk te weinig van IT om daar iets mee te doen. Dus het is heel lastig om daarmee te schakelen.
P: Dan moet je inderdaad. Goed over na kunnen denken om zo'n probleem systematisch aan te kunnen pakken.
I: Ja. Dat je kan elimineren van dit werkt wel en dat werkt niet. Hoe vind je daar de gouden middenweg in die voor beide werkt.
P: Wat vind je zelf van de manier waarop hier gewerkt wordt? Vind je dat prettig of zou je toch graag wat aanpassingen zien?
I: We zijn eigenlijk heel druk aan het aanpassen. Ik denk dat je maar zoveel

aanpassingen tegelijkertijd kunt doen.
P: Natuurlijk.
I: Het is ook een bepaalde mindset die je in je hoofd moet hebben om met die sprints te werken. Die items die er nu in komen. Die komen er niet voor niets in. Die komen er nu in omdat de mensen die ze er inzetten niet die mindset hebben. Die willen dat misschien niet. Die willen meer flexibiliteit bieden richting de klant. En eigenlijk zouden die sprints voor zekerheid moeten zorgen van wanneer er een release uitkomt. Dat is een beetje een wig waar we continue een beetje tussen schommelen. We zijn er dus super druk mee bezig en ik denk ook dat we de goede kant op gaan.
P: Dat is ook uitproberen van wat prettig werkt en op die manier ga je dan verder?
I: We proberen alle neuzen dezelfde kant op te krijgen. We doen een stapje vooruit, dan laten we aan iedereen zien van dit werkt toch ook veel fijner. We hebben nu elke week een nieuwe patch versie met kleine problemen opgelost, bijvoorbeeld.
P: Ok,
I: Er zijn natuurlijk altijd dingen die beter kunnen.
P: Ja, zie je in je omgeving nog bij mensen die je kent die ook in de IT werken, dat er op dezelfde manier gewerkt wordt? Of zie je nog veel verschillen daarin?
I: Ik heb op een andere plek gewerkt zoals ik in het begin vertelde. Ik denk wel omdat we een kleine organisatie zijn dat er daarom minder volgens stramienen wordt gewerkt. Dat heeft voordelen dat we veel flexibeler zijn richting klanten. Als klanten echt een issue hebben dan kunnen we dat sneller voor ze oplossen. Aan de andere kant heeft het ook nadelen. Dat er sneller bugs in de software voorkomen omdat er minder kwaliteitscontrole plaatsvindt. Ik heb wel eens bij bedrijven gewerkt waar dedicated mensen aan het software testen waren. Dat is hier absoluut niet, hier doen mensen het er meer bij. Het testen.
P: Zijn er hier ook mensen die ervoor zorgen dat algemeen de architectuur van het systeem op orde blijft en dat geen chaos wordt? Of doen jullie dat als ontwikkelaars er ook bij?
I: De code kwaliteitscontrole bedoel je?
P: Ja.
I: Gebeurt niet structureel. Ik denk dat iedereen een beetje in de gaten houdt wat er op het versiebeheer systeem wordt toegevoegd. Iedereen is er ook wel een beetje zelf verantwoordelijk voor. Iedereen die wat commit, moet er zelf achter staan. Er wordt niet echt. Ik denk dat net als ik net zei, dat mensen wanneer ze vragen hebben omdat ze ergens over twijfelen. Moet ik dat hier wel toevoegen of moet ik dat wel zo doen, dan wordt dat wel besproken. Iemand zou bij wijze van spreken zomaar iets kunnen toevoegen, zonder dat andere mensen dat zouden merken. Dat zou kunnen gebeuren.
P: Het is dus iedereen doet zijn eigen ding en als hij hulpt nodig heeft dan

vraagt hij dat. Anders gaat hij gewoon zelf verder?
I: Ik houd wel een beetje in de gaten van wat er langs komt maar ik kan niet alle code van vier collega's in de gaten houden. Want dan werkt ik zelf ook de halve dag niet.
P: Nee ik snap het maar..
I: Ja
P: Dit was het, bedankt.

Appendix C

Interview 3

Interviewer wordt weergegeven als P.
Interviewde wordt weergegeven als I.
P: Goedemiddag
I: Goedemiddag
P: Zou je wat kunnen vertellen. Wat je doet binnen het bedrijf, wie je bent?
I: Ik ben I we zijn hier bij bedrijf A. We maken enquete software voor sensorisch onderzoek. Dit doen wij via mobile devices, de browser en meestal hebben we een panel dat uitgenodigd moet worden. Dat betekend dat mensen op basis van een selectie uitgenodigd worden. Mannetje/Vrouwtje. Tussen bepaalde leeftijden. Maar omdat het ook over sensorisch gaat en dus voedsel, kan het ook allergien zijn. Voorkeuren, voorgaande resultaten etc. uiteindelijk als de enquete is afgenomen, dan analyseer je de enquete met onze analytische software en daar komt dan geijkte statistieken komen daar dan uit. Ik zelf ben hier gewoon software ontwikkelaar. Primair frontend maar opzich is het hier niet hierarchisch dus ik heb ook vaak zat dat ik backend dingen moet doen.
P: En zit je dan meer richting mobile gaat of meer richting web of toch echt meer van native applicaties.
I:We hebben voor de enquete software hebben we twee platformen. We hebben de browser based oplossing dus dat je gewoon eigenlijk op elk device met een redelijk moderne browser de enquete af kan nemen. Dus dan spreken we over een IE8+ en in principe alle andere browsers safari browser, ios safari, android chrome, je kent het wel. De andere kant is de standalone applicatie die op de mobile device geinstalleerd kan worden. We maken gebruik dan van titanium dus dat is een oplossing zodat we een keer iets hoeven te ontwikkelen en dan kunnen we het voor zowel android als iOS uitbrengen. Het voordeel van deze app ten opzichte van de browser is oplossing is. Dat de app alleen eenmalig om de enquete binnen te halen en eenmalig om het resultaat op te sturen verbinding maken. Dus dat betekend dat je niet meer altijd online hoeft te zijn. Dat heeft natuurlijk als voordelen

dat je veel meer dingen kunt doen, gebaseerd op offline bezigheden. Maar naast dat als je een app heb je ook locatievoorzieningen, je kan gebruik maken van een foto, een video, spraak. Dat soort dingen en als pilot, eigenlijk als prototype hebben we vorige zomer op de beurs laten zien dat wanneer we gebruik maken van google cardboard. Dus dat is dat google VR ding. Dat je dan het mobiele apparaat op je neus zet en dan hadden wij de applicatie gemaakt dat je de enquete vragen ja en nee bijvoorbeeld kon invullen met het schudden van het hoofd. Dat de ja en nee. Dus dan heb je een uniek iets dat echt alleen in een mobiel apparaat kan. Heb je misbruikt om een alternative flow te maken.
P: Als we dan gaan kijken naar het ontwikkelen van de software. Werken jullie daarmee op een Agile manier of toch wat meer waterflow?
I: Van nature is het eigenlijk opgezet met waterflow. Dat is van tevoren opstellen er doorheen jassen en dan kijken waar we stranden. We zijn steeds meer bezig om echt een iteratief proces te hebben. Tegenwoordig hebben we sprints, die ook een vaste duratie hebben. Dus in de overgangs periode van een sprint twee drie maanden duurde. Daar zijn we nu een stuk strikter op. Al is dat wel nog steeds met de nodige kanttekeningen. We hebben een wekelijkse patch ronde. Dus wanneer er kleine bug fixes gemaakt kunnen worden. Dan worden die in de patch meegenomen. Dan kunnen we dus in theorie elke week een nieuwe versie uitbrengen. Waarbij deze dingetjes opgelost zijn. In de praktijk komt dat er opneer dat als de klant een bepaald probleem heeft dan zullen wij voor die plant specifiek die versie updaten. Op het eind van de week, als de versie beschikbaar is.
P: Houd dat in dat een sprint bij jullie een week duurt?
I:Nee dat is speciefiek deze. Volgens mij volgens de officiele sprintregels duurt een sprint tussen de 3 en de 5 weken. Mja het is een stuk convienienter om voor deze patch methodiek, dat de sprint in een week laten plaatsvinden. We hebben de vast patchdag dinsdag. Op maandag worden alle patches in die sprint gezet, het voorbereidende werk. Uitloop hebben we op woendag. Op donderdag is de test en vrijdag gaat de hamerslag erop en zou er een versie moeten zijn. Maar goed ook daar zitten soms problemen, sommige issues duren gewoon langer en kan het best zijn dat een patchweek een dubbele patchweek wordt. Maar goed dat is een proces waar je niet echt aan ontkomt.
P: Nee dat klopt. En als er dan zo nieuwe functionaliteit dat, dat de eerste dag begint met het ontwerpen maken. Wat voor ontwerpen moet ik me daar dan bij voorstellen?
I: Het is vrij plat. We hebben geen specifieke persoon die deze functionaliteiten allemaal omschrijft en beheert, ja beheert wel. Dat gebeurt natuurlijk op de maandag van tevoren en achteraf op de vrijdag wordt gekeken of het allemaal klopt. Maar we hebben niet een speciefiek traject. Het is in principe bij de patch zo, wanneer een klant een probleem signaleert dan wordt door onze support afdeling een issue ingeschoten, daar wordt dan

een ticket omschrijving, vaak met een screenshotje of een gifje. Dus een screen capture in de vorm van een gifje daarbij gedaan. Als voorbeeld. En dan zou je op basis van die informatie, deels op eigen inzicht maar ook deels gewoon op de dag zelf langs moeten gaan bij support, om te kijken van wat is nou daadwerkelijk de correcte oplossing. En is de oplossing die ik denk dat er moet komen, de werkelijke oplossing. Dat is een proces waar nog wel wat fine tuning kan.
P: Als jij dan bij de support afdeling langsgaat, maak je dan ook nog voor jezelf tekeningen of zet je iets op papier of is het vooral dat je het in je hoofd doet?
I: In mijn hoofd en wanneer dat gebeurt is het zo dat ik vaak op de Jira issue erbij zet van zo hebben we het besloten. Zo dat het wel uiteindelijk terug te lezen valt. Het is niet heel erg formeel.
P: En als we dan gaan kijken naar het ontwikkelen van nieuwe functionaliteiten, hoe gaat dat dan in zijn werk?
I: Nou dan hebben we duidelijk de oude stempel. Die zeggen we beginnen gewoon met maken en zien wel waar het strand. Dat is dus bij sommige subprojecten zo, er zijn een paar grote lijnen die uitgelegd zijn en die thematisch bepaalde problemen moeten oplossen. En enkele van die themas die komen terug in een sprint van een lange duratie. Dus we hebben die wekelijkse sprint, maar we hebben ook de grote generieke sprint, dus gewoon van onze huidige development versie. En daarin zitten vaak die issues die bij vlagen niet goed neergezet zijn. Die eigenlijk helemaal niet uitgedacht zijn. Een tekort aan informatie. Maar je hebt ook weer aan de andere kant gevallen dat het helemaal duidelijk is. Dat er eventueel zelfs een design bij zit. Implementatie specifieke dingen. Van te voren is het sinds de laatste drie maanden is het dat voor de sprint begint. Moet ontwikkelaars gevraagd worden, hoe lang ga je erover doen. Dus dan kan er een reele tijdsplanning inkomen. Op basis van dat kun je ook een reele periode afspreken dat de versie af moet komen. Met de gevraagde functionaliteiten.
P:Voor die functionaliteiten maak je dan nog wel gebruik van iets van flow charts of business logic/ bedrijfsregels?
I: Nee over het algemeen is dat heel karig. Maar dan moet je er ook rekening mee houden dat het allemaal overhead is. Jij zelf als ontwikkelaar bent overhead. In principe het enige dat je wilt is een functie en wanneer het met minder af is en nog goed duidelijk is. Is het naar mijn mening een flow chart overbodig.
P: Die grote lijnen hoe worden die dan uitgezet? Wordt dat mondeling gedaan?
I: Die projecten zijn gedocumenteert. Gewoon wat er in moet komen, de ideeen. Er zit soms een kleine prototype bij, maar meestal is het gewoon een document die uitgewerkt is. Van wil je eigenlijk in deze functionaliteit hebben en inpricipe zijn deze deze dingen eigenlijk altijd funcitonal requirements. Non-functionals zitten er nauwelijks in. Dus dat

heeft als probleem dat als non-functional zou je onderhoudbaarheid heel er hoog in het vaandel kunnen stellen omdat wij continu bezig zijn aan een software suite. Maar dat ontbreekt dus.
P: Is dat dan ook nog een beetje richting use cases, van wat een functionaliteit moet gaan doen? Die jullie neer zetten.
I: Nee, heel sporadisch zit er een email conversatie bij van wat de klant verwacht. Er worden geen use cases gemaakt. Ook geen test op users. In principe wordt het bij een specifieke klant neergezet. Die dan in principe stakeholder is voor dit idee. Je hebt een bepaalde klant die zegt ik wil een plaatje hebben waarop ik kan klikken. Als er dan niet specifiek gedefinieerd is, dan krijgt de klant een plaatje waarop hij kan klikken. En dan is het maar aan de klant om te zeggen, is dit wat je zocht.
P: Dus dan is het echt dit moet erin dat moet eruit. De functionaliteit puur op basis van input/output, dat moet het gaan doen?
I: Ja, Ja. Je zou in theorie kunnen zeggen dat de eerste gebruiker ook meteen de tester is.
P: Ok, wat vindt jezelf van deze manier van werken? Is dat prettig of zouden bepaalde dingen ook wel anders ?
I: Je hebt natuurlijk klanten en je hebt medewerkers van die klant. Vaak is het dat een medewerker van een klant iets wil, of een groep van medewerkers van die klant die iets willen. Dan krijg je dus eigenlijk dat je niet zozeer een product in een versie erbij zet, maar dat je zegt we gaan dit met jullie proberen en dan komt daar maar iets uit. Vanuit daar ga je zeggen dit is leuk maar we willen dit erbij of dan heeft een andere klant die zegt: oh dat is ook wel een goed idee. En dan breiden we er dan op uit. Het nadeel daarvan is je vaak twee/drie keer aan hetzelfde stuk. Gewoon puur omdat er af en toe best grote wijzigen nodig om het flexibel genoeg te maken, voor meerdere klanten tegelijk praktisch werkbaar te maken.
P: Is dat dan het gevolg van dat er te snel aan begonnen is? Of is dat een gevolg van wat jullie doen?
I: De combinatie natuurlijk, je hebt de bedrijfscultuur dat er graag gelijk begonnen wil worden. Maar daarnaast heb je ook de druk van de klant erop gaat zitten. We verwachten dat dit geleverd wordt. En vaak wilt er ook nog wel een periode overheen gaan voordat er ingezet wordt. Het verzoek erin gezet wordt. Dan komt de tijd als probleem ophoog zetten. Dan komt door tijdsnoods dat je eigenlijk daar niet meer genoeg tijd aan kunt besteden en waardoor het later wellicht meer tijd kost. Maar het wel een partieel product heeft opgeleverd.
P: Is dat een manier wat als je daar aan werkt dat, dat prettig werkt?
I: Het mooiste zou zijn als ontwikkelaar is dat je eigen inbreng is goed. Maar het moet wel op het juiste moment zijn. Ik ben van mening dat van tevoren vaak gebeld kan worden en daar dan op gereageert kan worden. Dan zou er dus daarop het eind resultaat veranderd kunnen worden. Het probleem is alleen, die stap zit daar niet in. Op het moment dat je iets moet gaan

bouwen dan is het al dat je het moet gaan bouwen. Dus dat je eigenlijk dezelfde dag of dezelfde periode moet aan dit stuk moet gaan werken en dat er eigenlijk geen moment is om hier echt goed over na te denken. Je verwacht als ontwikkelaar dat je een issue krijgt, dat je implementatie klaar iets krijgt. Waar eigenlijk minimaal over nagedacht moet worden. Want het grote nadenken, de lijnen moeten er al uitgelegt zijn. Vaak missen die lijnen dan. Dan kom je op een implementatie en dat moet je eerst nadenken of jou ideeen over die implementatie wel goed is. Dan komt ook vaak zat voor dat je niet voldoende hebt gecommuniceert daarover en dat er een deel gescratchet moet worden.
P: Dat is zonde.
I: En frustrerend vooral.
P: Dat kan ik me voorstellen. Als je om je heen kijkt bij andere mensen die hetzelfde doen. Zie je dan dat daar op dezelfde manier gewerkt wordt of zie je daar andere manieren waarop gewerkt wordt?
I: Dat hangt heel erg af van de grote. Bij grotere bedrijven zul je heel snel zien dat dit soort dingen heel erg strak geregeld zijn, omdat het anders niet behapbaar is of dat er niemand verantwoorbaar is. Dan zul je gelijk zien dat het allemaal van te voren vastgelegt is, of dat er een andere expert overheen kijkt. Dat er meerdere experts gewoon naar kijken. En dan komt er een oplossinng uit en die wordt dan gebouwd. Nu is het bij ons zo, we hebben een idee. Er is misschien een oplossing. We gaan wel kijken of het lukt. Er zit natuurlijk groot verschil in, je hebt heel veel bedrijven die zeggen we beginnen gewoon. Maar dat heeft zijn gevaren.
P: Dan wil ik je hartelijk bedanken.
I: Ok.

Appendix D

Interview 4

Interviewer wordt weergegeven als P. Interviewde wordt weergegeven als I. P: ten eerste kun je wat over jezelf vertellen? Wat je doet binnen de organisatie? Wie je bent?
I: Ok. Mijn werkgever is bedrijf A. Een vrij kleine it staffer uit Nieuwegein. Daar ben ik aangenomen, pas geleden. Als senior content developer. Dat betekend dat ik met name alles tussen de server tot aan de browser programeer. Niet zozeer ontwerp maar wel code schrijf. Dat heb ik de eerste poos gewoon voor interne projecten gedaan. Sinds een poosje zit ik gedetacheerd bij bedrijf B. Om daar een claims applicatie voor alle labels van bedrijf A te schrijven. Dat is wat ik nu aan het doen ben.
P: En als je zegt dat je een senior frontend developer bent. Heb je dan ook nog mensen onder je?
I: Bij bedrijf A heb ik 1 iemand onder me. Dat worden er over een maand, dan komt er nog 1 iemand bij. Bij Open Web zelf niet, maar ik wordt zeg maar gedetacheerd als senior man.
P: Bij het ontwikkelen van applicaties dan gaat het zoals je net aangeeft over die labels. Dat je een functionaliteit ontwikkel. Worden er dan gebruik gemaakt van modellen of iets van tekeningen?
I: Nou ja bij bedrijf B wordt het wel heel strak gedaan. Eerst worden er requirements geschreven. Op basis daarvan worden use cases geschreven. Op basis daarvan gaan de systeemachritecten en functioneel ontwerpers, die gaan functioneel ontwerpen maken. Dat gaat veel gepaart met diagrammen, die ik niet altijd nodig vindt. Op basis van die functioneel ontwerpen gaan de frontend en backend ontwikkelaars aan de slag en in dat proces zitten we nu.
P: Als je vanaf de frontend hoe dingen er uit moeten zien is dat allemaal beschreven?
I: De ontwerpen zijn gemaakt door een extern bureau. Door webdesigners. Zeg maar hoe het er visueel uit moet komen te zien. Wat ik doe is ervoor zorgen dat het ook daadwerkelijk werkt. Dat het niet alleen platte html is

maar dat alle functionaliteiten ook werkt. Daarom bouwen we nu een vrij grote angular applicaties nu. De backend lui die richten de API's in die we daarvoor gebruiken. De frontend lui, dat is ik en die andere jongen dus. Wij zorgen dat alles functioneel in de browser werkt.
P: Je geeft net aan dat bij het functioneel ontwerp je niet alles nuttig vindt of overbodig. Wat vind je dan niet nuttig of overbodig?
I: Ik vind dat er wel erg veel van te voren in stramienen vastgelegd wordt. Waardoor je wel erg rigide moet programeren. Soms denk ik bij mezelf, het is allemaal leuk en aardig wat er in zo'n functioneel ontwerp staat. Maar het is omslachtig en het hoeft niet zo omslachtig. Maar goed dan bel je een functioneel ontwerper en dan zeg je hoi, hebben jullie er niet aan gedacht om het zo te doen? Ja dat kan ook wel of hij zegt nee het moet zo want, dat is onze standaard in de enterprise archtectuur. En dat zei dan dat zo, maar het ligt soms wel erg strak vast. Dat is opzicht niet erg, dat maakt het ook wel makkelijk, mijn baan. Maar het maakt het ook heel inflexibel en dat maakt het soms wel jammer.
P: Zie je dan nog wel voordelen aan deze manier van werken?
I: Ja absuluut, het is. Zeker als het gaat om een boel geld en daar gaat het gewoon om. Kun je van voren vrij makkelijk ramen van hoe lang iets gaat duren. Het hele process wordt een stuk voorspelbaarder dan als je maar op de bonne fooi gaat zeggen. We willen een claims applicatie, veel plezier ga je gang. Dus dat is opzicht wel goed en dat maakt, dat ik een stuk meer houdvast heb. Maar het is af en toe gewoon te rigide.
P: Ok, Je geeft nu aan dat dit in het geval van bedrijf A zo is. Heb je ook nog ervaringen van bij andere organisaties van dat het dan heel anders gaat?
I: Ja dat heb ik wel. Bij de Bedrijf C waar ik vroeger gewerkt heb. Daar was het wel echt zo van, we willen dat het ongeveer dit doet. Zorg maar dat je het oplost. Nou daar heb ik ook vaak deadlines overschreden. Dat had daar misschien ook wel mee te maken.
P: Dus, daar werd veel minder gebruik gemaakt van tevoren dingen bedenken.
I: Ja!.
P: Heb je nog meer voorbeelden?
I: Nja opzich, ik heb voor Open Web ook een maand gewerkt aan een ander project. Daar werd wel gebruik gemaakt van een issue tracker en daar werd allemaal netjes in geschreven, maar de product owner die besloot steeds halverwege een sprint van nee ik wil het toch anders. Dat druist tegen alle principes van het agile werken in. En dat maakt het dus best onhaalbaar om sprints te halen. Dus dat is eigenlijk niet zo fijn. Het is heel fijn als van tevoren de user stories en de sprint deliverables vast liggen. Maar als je er al te rigide aan vasthoud, waar ze bij bedrijf A een handje hebben om dat te doen. Dan ga je ook weer het hele doel van agile ontwikkeling voorbij. Dat je flexibel ook bent.
P: Want het is allemaal Agile ontwikkeling wat jullie doen?

I: Ja.
P: Een sprint, hoe lang duurt die dan ongeveer?
I: twee weken.
P: Heb je nog ervaring met andere tijden, of is dit gewoon
I: Nee, waar ik mee gewerkt heb, was het altijd twee weken. Je kan ook zeggen dat je het een week doet of drie werken. Maar twee weken is ..
P: Dat wordt het prettigs ervaren om alles van zowel ontwerpen als functionaleiten..
I: Ja precies. Vooral als je een sprint wissel hebt. Ben je echt een halve dag mee bezig met de volgende sprint te bepalen en als je sprint maar een week is, heb je daarna eigenlijk nog maar 4.5/4 dagen om te ontwikkelen. Voordat je weer tegen een sprint wissel aanzit. En dat is gewoon niet zo heel veel. Dan kan je geen echt complexe dingen tackelen, in 1 sprint.
P: Als je een ontwerp hebt van wat jij krijgt. Maak je dan ook voor jezelf nog wel eens een klein modelletje van zo wil ik iets gaan doen? Of tekeningetjes?
I: Ja. Ik uh. Al mijn collega's werken met visio. Dat flow programma van Microsoft. Ik kan er niet mee overweg. Ik veel te veel tijd bezig met de goede vormpjes uitzoeken. Dus ik sta geregeld voor ons whiteboard in onze projectruimte te krassen en te schrijven en te schetsen van hoe wil ik precies dat bepaalde klasse met elkaar communiceren en moet dat daar een super overheen en dat soort zaken.
P: Gewoon dat je soort van UML maakt?
I: Ja dat is inderdaad UML charts. Die worden veel gebruikt bij bedrijf B. Maar ik kan gewoon niet zo goed overweg met fysio.
P: Daarom doe je hetzelfde op papier.
I: Ja, daar komt het wel op neer. Maar dat komt ook omdat ik misschien een keer een dag tijd moet steken in visio. Om visio te leren, want zo moeilijk kan het niet zijn.
P: Dan is het ook wat mensen fijn vinden of om het op papier te doen of niet op papier te doen.
I: Ja, dat is waar.

Appendix E

Interview 5

Interviewer wordt weergegeven als P.
Interviewde wordt weergegeven als I.
P: Welkom, ik wil je in iedergeval alvast bedanken voor het interview. Kun je wat over jezelf vertellen? Wie je bent? Wat je Doet?
I: Dat is goed, mijn naam is I. Ik ben informatiemanager voor bedrijf A. Dat is een faculteit van de universiteit . Ik doe dit werk vrij lang. Ben begonnen als studentassisent en altijd hier gewerkt. Ik heb qua opleiding eerst Theologie gedaan en later overgestapt op informatica. Die heb ik in deeltijd afgerond en ik ben afgestudeerd op migratie trajecten.
P: Op het moment dat jullie een applicatie nodig hebben en die ontwikkelen. Hoe gaat zoiets dan te werk?
I: We zijn een vrij kleine faculteit. Voor ons is het een soort aardig bijproduct. In principe proberen we natuurlijk eerst te kijken naar wat er al binnen de universiteit is ontwikkeld aan programmatuur, maar in de praktijk betekend dat vaak toch dat we of wel dingen in eigen beheer ontwikkelen ofwel in combinatie met het bedrijf dat we inhuren om het te doen. Het ontwerp wordt wel meestal in huis gedaan en ja de implementatie dat hangt er een beetje vanaf.
P: Als je zegt dat het ontwerp wordt binnenhuis gedaan. Wat valt er dan onder dat ontwerp?
I: Eerst natuurlijk nagaan wat er precies nodig is. Proberen het projectmatig aan te pakken. Dus eerst de requirements vast te stellen en dat komt dat in de project documentatie. Maar het gebeurt ook regelmatig dat dingen informeel gebeuren en dan proberen we eigenlijk zo te werken dat we zo snel mogelijk een soort prototype hebben wat we kunnen laten zien. Dat we daarna met de gebruikers samen ernaar kijken van wat is hier goed aan, wat moet er hier anders. Of uberhaupt gaat dit wat worden. Dus ook een soort haalbaarheidsstudie te doen eigenlijk.
P: Ok, en als je zegt dat je dingen informeel doorneemt? Dat is gewoon puur doorspreken of?

I: Ja, het punt is vaak dat je vaak als ontwikkelaar en ontwerper. Het terein waar je voor ontwikkeld goed kent of denkt goed te kennen. Dus dan denk je dat je alles in huis hebt om het te kunnen ontwerpen. Maar goed de taakverdeling, omdat er een soort opdrachtgever is die bepaald of het de goede kant uitgaat en een andere de bouw kant en de ontwerp kant om die toch goed te scheiden is toch. Wat ik wel heb ontdekt dat heel belangrijk is.
P: Hoe probeer je dat dan te doen?
I: Zoveel mogelijk de gebruiker erbij te betrekken en de rol van opdrachtgever te geven. Binnen de universiteit heb je niet natuurlijke rol die mensen hebben. Mensen komen bij me met een bepaald probleem en ik denk dan snel van, ik weet wel een oplossing, maar toch die rolverdeling is toch wel belangrijk om uiteindelijk een goed product te krijgen.
P: Dus als de gebruiker het goed vindt dan is het goed?
I: Dan is het goed.
P: Ok en bij dat ontwerpen maak je dan ook nog gebruik van tekeningen of ?
I: Nja ik heb wel eens geprobeerd om consequent alles eerst in user stories onder te brengen en op grond daarvan te gaan ontwikkelen. Of een soort mock-up te maken van tevoren. Dat je een soort interface kan laten zien van zo en zo, hoe het eruit gaat zien. Dat kost ook aardig wat tijd. Het is eigenlijk wel rendabel maar dat is altijd een beetje van een spanningsveld. Van neem je de tijd en de moeite om dat van te voren goed te documenten, UML diagrammen te maken zodanig dat je er zelf wat aan hebt als ontwerper, maar het mooiste is als je het zo kunt maken dat de toekomstig gebruiker daar ook in zit en inzicht krijgt hoe je het voorplan bent te gaan doen. Voordat je het daarwerkelijk gaat doen.
P: Ok. Als je zegt die mockups probeer ik te maken,maar je vind ze ook heel erg handig.
I: Ja eigenlijk wel. Je hebt de neiging om die stap over te slaan. Terwijl je jezelf dan toch later tegenkomt. Dus het is een vorm van dicipline om dat toch te doen. Zoveel mogelijk.
P: Heb je daarin dan nog vaste stappen van als ik een project ontwerp ga ik gewoon deze stappen achter elkaar af en dat doe ik consequent om mij daar aan te houden? Of moet je jezelf er aan blijven herinneren van hier moet ik dit doen en daar dat?
I: Bij de wat kleinere klussen heb je de neiging om heel veel stappen over te slaan. Dat je denkt van acht ik weet wel hoe dat moet inmiddels. Zo keer je een beetje terug naar de ongestructureerde manier van werken. Die je zou doen als je niet beter wist.
P: Het lijkt erop alsof het heel makkelijk is om even snel dingen te doen en discipline te houden, in het ontwerpen. Ondanks dat je het gevoel hebt dat het heel duidelijk werk.
I: Ja! dat is een soort leerproces. Om dat toch toe te passen omdat het allemaal zo kleinschalig is. Dat niemand je op de vingers tikt. Zo van je

vergeet nu die stap of een andere stap. Want eigenlijk moet je jezelf eigenlijk ook je daarin managen dat je niet alleen bezig bent als ontwikkelaar maar ook, de rol van project manager eigenlijk op je neemt en dat je die rollen een beetje probeert te scheiden. Dat valt natuurlijk niet mee als je dezelfde persoon bent.
P: Ja en maak je dan ook nog gebruik van Business logic bij het ontwikkelen of laat je dat vanuit de gebruikerseisen komen, of weet je dat zelf?
I: Vaak weet je dat zelf wel, maar het is wel goed om dat bij de gebruiker goed te toetsen. Ook omdat je vaak denkt dat je weet wat de gebruiker wilt. Dat kan soms voordelig zijn, want dan heb je gewoon gelijk. Maar het komt ook regelmatig voor dat je gaandeweg ontdekt, van wacht ik dacht dit maar ja het zit toch anders in elkaar.
P: Probeer je dat nog op een gestructureerde manier te doorlopen? Om ervoor te zorgen dat je de goede dingen eruit haalt? Of is het eerstse versie werkt niet, we gaan naar een tweede versie en lossen het daarin op?
I: Ja zo gaat dat, het is eigenlijk de terugkoppeling krijg je toch wel. Maar het is goed om die terugkoppeling wel gestructureerd te houden en ieder moment te weten van ben je op de goede weg of niet.
P: Ok en wat vind jezelf van deze manier van werken? Of zou je graag toch nog iets daarin willen veranderen?
I: Iets meer structuur is toch wel goed. Neem een onderwerp als documentatie van alleen het ontwerp proces. Maar ook van documentatie van de gebruikershandleiding. Omdat je de gebruiker meeneemt in het proces. Dan weet de gebruiker op een gegeven moment ook heel goed hoe het werkt. Maar met het risico dat je het niet documenteerd en dat geld niet alleen voor het ontwerpen van programatuur maar eigenlijk ook bij installatie klussen. Het is heel belangrijk dat je alle stappen die je daarin doet te documenteren. Ja dan merk je soms toch dat je bepaald stukjes toch net niet hebt gedocumenteerd. Als je dat nodig hebt kom je erachter. Het is toch fijner om dat compleet te hebben.
P: Ik hoor je vaak over dat het documenteren heel goed moet gebeuren. Maar je hebt ook software ontwikkelmethodes waarin meer documentatie er op wijst dat je niet goed bezig bent. Maar ben je het daar mee eens? Of zeg je dit is wat hier goed werkt?
I: Ja, natuurlijk is er een balans. Je kunt docoumentatie ook gebruiken alle kromme en handige die in het systeem zitten te documenteren. Inplaats van ze op te lossen. Hetzelfde geld bij het documenteren als je code schrijft. Schrijf je code zo helder dat je geen commentaar nodig hebt. Of ga je, je code leesbaar te maken door bij iedere module te gaan noteren wat doet het en hoe werkt het. Of kies je ervoor om dat met behulp van testen te doen. Dat is opzich wel weer een betere methode en ik denk opzicht wel dat het goed is om documentatie wel compleet te hebben maar wel minimaal. In die zin van het moet wel helder zijn maar geen eigen project van maken om het geheel te documenteren. Het moet wel functioneel zijn.

P: Je zegt dat deze manier van werken hier erg prettig is, maar ben je ergens anders dat je dingen gezien hebt waarvan je zegt dat zou ik ook nog wel eens willen doen?
I: Ik heb wel eens geprobeert om volgens het principes van Price2 te werken. Ik heb er ook aardig wat dingen over gelezen en wat dingen uitgeprobeert. Er zitten aardig wat goede dingen in. Ook in de structuur die je dan hebt. Ik zou eigenlijk wel een combinatie willen maken van Prince2 met Agile. Dat je en goed documenteert en structureert maar dat je ook wel de flexibiliteit houdt.
P: Je zegt dat je graag een combinatie met Agile wilt maken en agile werkt op basis van scrums. Maar je dan ook gebruik van vaste tijdseenheden waarin je wilt ontwikkelen of is die scrum lengte flexibeler.
I: Die lengte moet wel flexibeler zijn want, de tijd die we aan projecten kunnen besteden is ook afhankelijk van wat er verder gebeurt omdat we ook service en support moeten leveren. En er andere projecten er doorheen lopen, die eigenlijk niet zoveel met het ontwikkelen van software te maken hebben. Maar met de implementatie van dingen die binnen de universiteit spelen. Ja, dat kan heel erg schuiven en daardoor moet je de projecten een beetje door elkaar heen vlechten. Zo blijven sommige projecten ook heel lang liggen.
P: Dat probeer je eigenlijk door agile te ontwikkelen te voorkomen Maar dat kan niet anders?
I: Ja dat kan niet anders.
P: Ok, Heb je ook nog ervaringen van dat je bij andere bedrijven ziet hoe daar dingen gaan? Als je dan iets outsourced dat je dan ziet dat het team ook wat aan ontwerpen doen of ..?
I: Ja dat heeft inderdaad erg geholpen. Met het werken met sprints. Het wordt allemaal wat concreter. Ik had het wel op congressen enzo meegemaakt. Hoe je met sprints aan open
source producten werkt. Maar dat het ook bij commerciele zaken gebruikt wordt, dat is ook wel een eye opener. Het bedrijf dat we zelf inschakelen, wordt de eigenaar ook ingehuurt bij KPN om daar de SCRUM methodologie te leren. Dus dat gaf de burger wel moet. Om te zien dat het een geaccepteerde methode wordt en geen hobbyisme verschaft.
P: Omdat je zelf op een kleine afdeling werkt, dat je niet altijd ziet hoe het buiten gaat?
I: Ja inderdaad, precies.
P: Ik wil je bedanken.

Appendix F

Interview 6

Interviewer wordt weergegeven als P.
Interviewde wordt weergegeven als I.
I: Oh je was nog niet begonen?
P: Nee, Ik zal je eerst even wat algemene vragen over jou om een idee te krijgen wat jij zoal doet. Zou je over jezelf iets kort kunnen vertellen? Van wie je bent en wat jij bij de bedrijf A doet?
I: Ik heb zelf scheikunde gestuurd aan de universiteit van Leiden en tijdens een stage in cambridge ben ik in aanraking gekomen met het internet, webpagina's en websites. Dat vond ik gaaf, dit was in 1996. Toen ben ik eigenlijk daarmee aan de slag gegaan voor mijn studentenhuis waar ik toen woonde, voor de studentenvereniging waar ik lid van was. Hun websites maken en daar databases aan koppelen zodat je dingen wat makkelijker kon doen en dat was allemaal met PHP en javascript en html natuurlijk. Zo ben ik langzaam op geschoven naar professioneel websites bouwen voor grote enterprises. Ergens heb ik java opgepikt als programeer taal. Ook omdat ik altijd naar java keek als echte programeertaal en javascript meer als spielerei. Maar je ziet nu dat javascript terug aan het komen is als serieuze ontwikkeltaal. Dus daar verdiep ik me ook al jaren weer in. Ik begon niet bij de bedrijf A, ik werk nu twee jaar bij de bedrijf A als software ontwikkelaar voor het realisatie online, zoals ze dat noemen. Dat komt eigenlijk neer op de website. Op de website heb je /reisplanner, dit is waar ik op dit moment aan werk en een angularjs applicatie, met java.
P: Dus je bent nog steeds met websites bezig?
I: Ik ben nog steeds met websites bezig. Op de website zit ook een heel e-commerce platform. Dat zit achter /producten en zo hebben we diverse applicaties in beheer. Wat we ook hebben, is het onboard information system. Als je in de trein zit heb je van die blauwe schermen waar op staat waar je bent en hoelang het voordat je aankomt. Dat is een webapplicatie die maken wij. Als je op bezoek gaat bij attracties in nederland hebben ze soms van vertreksstaartje van dichtsbijzijnde station, dat is webapplicatie.

Die maken wij en zo heb je nog een aantal dingen die je niet direct assosieert met web maar wat wel een heel makkelijk via het web ontsloten kan worden. We hebben ook de private API, dus als jij graag een applicatie wilt maken met treintijden dan kan je bij ons een API sleutel aanvragen en dan kun je daar zelf mee aan de slag. De private API is een java backend en dat bouwen wij ook.
P: Dat is zodat Google en 9292 hun informatie ook kunnen krijgen?
I: Ja, inderdaad. Een andere bekende is rijden de treinen dat is een app voor op android, geloof ik. Of in iedergeval voor op android.
P: Zo zijn heel veel producten..
I: Ja, inderdaad
P: En zoals in veel ontwikkel omgevingen zal het regelmatig voorkomen dat jullie met een nieuw onderdeel bezig gaan. Zou je mij kunnen uitleggen hoe zo'n cyclus van het maken van een nieuwe functionaliteit er ongeveer uit?
I: We werken volgens scrum/agile en dat begint al bij de ontwerpfase waarbij er vrij vroeg schetsen gemaakt worden en die worden dan aangehouden tegen de interne klant, voor wie wij het product aan het maken zijn. Maar ook al naar buiten toe, naar een klantenpanel en ook naar ontwikkelaars toe of dingen haalbaar zijn of niet haalbaar zijn.
P: Die schetsen zijn over hoe het er uit komt te zien of is dat om de functionaliteit wat te visualiseren?
I: Ja alles begint bij een business wens. De reisplanner even als voorbeeld. We willen de reizigers voorzien van reisinformatie en het moet dit kunnen en het moet dat kunnen en zus kunnen. Vervolgens proberen we heel snel toe te werken naar een eerste interne versie zodat mensen gaan zien. En geinspireerd worden door wat er gebouwd wordt. Omdat de ervaring is en dat is agile/scrum. Dat een werkend product altijd lijd tot nieuwe wensen en aanpassingen in bestaande wensen en dat we daar zo snel mogelijk mee aan de slag kunnen.
P:Want je ziet natuurlijk niet wat je allemaal wilt hebben.
I: Nee.
P:En dan maken jullie daarnaast ook nog bij het opstellen van dat soort dingen gebruik van bijvoorbeeld use cases of is dat gewoon meer jullie proberen iets en dan kijken of het werkt ?
I: Nee, we werken meer met user stories. Dat zijn net iets kleinere brokjes en dat lijdt uiteindelijk tot wat wij noemen eppics. Dat zijn grote brokken werk en dat splitsen we in kleine brokken werk. We werken in sprints van twee weken. Dus we willen graag dat iets in twee weken afgerond is.
P: en als het dan niet af is?
I: Dan schuift het door naar de volgende sprint. Tenzij dat de product owner, dat is de eigenaar van het product. Aangeeft dat de prioriteit verschoven is en dat het niet langer relevant is.
P: Of dat het te lang duurt?
I: Ja, maar dat gebeurt niet vaak. Je schat de dingen in qua punten. Dat

is een beetje een arbitraire schaal. Ik en mijn team hebben een velocity van 12 punten over twee weken. Dus als wij zeggen van we kunnen dit maken en dat kost twee punten, dan weet je globaal hoelang we er mee bezig zijn. En daarbij hoort dan gelijk al het test werk en ook de test automatisering en performance testen.
P: Dus jullie testen wel veel?
I:Ja we testen heel veel. De backend proberen we op 80/85% test coverage te hebben. Dan heb je het over unit test en een stukje intergratie tests. De test automatisering van de frontend proberen we functioneel op 100% te krijgen. Ja er wordt veel getest.
P: Ja, er wordt natuurlijk als mensen iets van bedrijf A zien dan denken ze dat moet werken. Grote organisatie.
I: Ja en doordat je snel itereert over je product heen is de kans op regressie groter. Dus hoe meer test je er tegenaan zet en dat is eigenlijk het vastleggen van je afspraken. Als ik hier op klik moet dat gebeuren. Is een afspraak die je met elkaar maakt. Wil je vastleggen en als je het niet zelf test is uiteindelijk wel een gebruiker die gaat testen of er gewoon mee aan de slag gaat. En die raakt dan geirriteerd.
P:Ja en die regels van wat iets moet doen. Dat is natuurlijk weer gekoppeld aan de bedrijfsregels van zo willen wij dat doen.
I: Ja.
P: Vind je deze manier van werken prettig of zou je er graag iets van een aanpassing in zien?
Dat je bijvoorbeeld in de praktijk merkt dat twee weken net iets te kort voor sprints of net iets te lang.
I: We hebben wel eens gespeeld met de periode. We hebben het ook wel eens op 3 weken gezet en dat beviel net iets minder. Twee weken is wel prima. Na twee weken leveren we ook daadwerkelijk een product op wat gereleased kan worden. Alleen het gebeurt nog niet altijd dat de product owner: Ja dit is goed, het kan naar productie.
P: Dan krijg je gewoon een nieuwe iteratie ?
I: Ja, en dat is zeker in het begin van zo'n reisplanner. We zijn er een jaar mee bezig geweest en na 1 jaar werd deze als beta naar buiten gebracht. Voor die tijd stond deze intern natuurlijk wel op allerlei plekken maar er was nog geen klant die hem had gezien.
P: Dat is natuurlijk ook weer heel spannend want klanten hebben ook weer een andere visie op hoe iets moet zijn.
I: Ja, ja. Het is wel zo dat wij. We hebben frontend ontwikkelaars. We hebben ook UX designers die dus de flow door je applicaties beschrijven en uitwerken, maar dat wordt ook getest. Daar huren we dan een bedrijf voor in, die gaat daarmee aan de slag. Die heeft een clubje mensen met wie zij door die designs heen gaan lopen en gaan kijken of mensen daadwerkelijk kunnen doen wat zij willen doen. Dan pas je daar uiteindelijk je design op aan en ga je in beta naar buiten. Dan komt de eerste feedback van klanten,

maar dat is ook allemaal heel summier. Pas als je de oude planner uitzet en dan de nieuwe planner aan. Dan komt de echte feedback, dan gaan mensen gedwongen gebruik maken van die applicatie.
P: En dat kan weer problemen opleveren voor mensen die tevreden waren met de oude applicatie?
I:Ja en dat is momenteel ook het geval met de reisplanner daar is heel veel feedback op gekomen van mensen die niet hun reizen kunnen plannen. Die niet begrijpen hoe die interface werkt en is vooral des te moeilijk. Op tablet en mobiel werkt het allemaal goed. Dat zien we ook terug in de gebruikers cijfers, we meten het. Maar op desktop valt het tegen. We zijn nu ook bezig om het design aan te passen en te zorgen dat mensen op desktop ook beter hun ding kunen doen. Dat is bij de reisplanner wel soort van een vergissing geweest. Wat opzicht wel bijzonder is want we hebben wel op allerlei mogelijke manieren feedback bevraagt, aan onze klanten. Maar dat is er toch niet uit gekomen, dat ze hun ding niet konden doen.
P: Dat is, dat komt of heel goed naar buiten of
I: Wat je nu ziet staan op /reisplanner gaat over een tijdje weer wat nieuws worden en we willen dan ook gaan A -¿ B testen. Dus gaan kijken van we voegen nu een feature toe, werkt dat? Helpt dat mensen om beter hun ding te doen of werkt het misschien beter als we de knop daar zetten of geel maken of blauw maken.
P: Ja daar zijn heel veel theorien over, inderdaad van wat werkt en werkt het in de praktijk net zo als het bedacht is. I: Ja, je kunt het dus ook prima meten. Je kunt bij wijze van spreken twee versies van die reisplanner naar buiten zetten. Eentje met een blauwe knop en eentje met een gele knop. Dan kun je gewoon meten wat beter werkt.
P: En dat is jullie voordeel, dat jullie daar controle over hebben. Van wat jullie naar buiten zetten en werkt iets niet dan kun je het ook weer relatief eenvoudig wijzigen.
I: Ja
P: Zie je verder in je omgeving dat er veel op dezelfde manier gewerkt wordt met het ontwikkelen van dat soort webapplicaties of zie heel veel verschillende dingen?
I: Nee, ik heb het idee dat de industrie voor het maken van webapplicaties redelijk werkt vanuit standaarden. Zeker als het gaat om grote bedrijven. Enterprise grade webapplications is wel redelijk hetzelfde. Ook qua technologie wat wordt ingezet. Zag je vroeger dat Java eigenlijk gebruikt werd voor het genereren van html en dat werd dan naar de client gestuurd en daar is iedereen een beetje van af aan het stappen. Gebruiken we nu veel meer JavaScript frameworks voor en dat je Java echt alleen nog maar op de server tegenkomt. Dus dat doet iedereen wel. Het is ook echt een ambacht wat wel redelijk uitgekristaliseert is onderdanks dat er veel dingen veranderen. Je hebt framework dit framework dat en dat..
P: En zeker met die mobiele applicaties dat vraagt natuurlijk ook weer

bepaalde innovaties.
I: Daar heb je inderdaad een manier van werken dat heet responsive webdesign voor. Dat je uitgaat van een mobiel apparaat en dat je webpagina daar goed op te zien moet zijn maar tegelijkertijd ook als je hem gaat schalen, dat die dan ook prima op een desktop te gebruiken is. Dat is ook zoiets dat doet iedereen.
P: Ja dat klopt.
I: Je ziet dat het gebruik van een apparte mobiele applicatie is verleden tijd.