De Thinkwise Lifecycle

Advies voor het duurzaam verbeteren van de software lifecycle.

Davey Brom

In opdracht van Thinkwise Software B.V.

Juni 2019

De Thinkwise Lifecycle

Advies voor het duurzaam verbeteren van de software lifecycle.

Door Davey Brom
Student Academie Creatieve Technologie
Business IT & Management

Klas DHI4VBAF
Student 426694

Begeleider: J.A. Schokker
Tweede beoordelaar: S.M. Versloot

In opdracht van M. van den Berg, namens Thinkwise Software B.V.

Apeldoorn, 17 juni 2019

Samenvatting

Dit onderzoek heeft plaatsgevonden bij Thinkwise Software B.V. Een software development bedrijf gevestigd in Apeldoorn. Al 17 jaar ontwikkelt Thinkwise Software haar eigen Low Code platform. Dit platform is vooral ontwikkeld om klanten een duurzame IT-oplossing te bieden en hen "Future-proof" te maken. Thinkwise is de laatste jaren sterk gegroeid waarbij de omvang van projecten ook is toegenomen. Naar aanleiding van een evaluatie van recente grote projecten is de conclusie getrokken dat er weinig inzicht is in de kwaliteit van het product tijdens de uitvoering van het project. Dit is pas duidelijk als het product wordt getest en opgeleverd. Dit moet duurzaam verbeterd worden zodat er kwaliteitscontroles in het proces worden opgenomen en er inzicht komt in het ontwikkelproces. Om dit zo volledig mogelijk te onderzoeken wordt dit onderzoek gericht op de volledige software lifecycle van Thinkwise. Hierbij is de volgende hoofdvraag geformuleerd:

"Hoe kan Thinkwise de kwaliteit van haar software lifecycle op een duurzame wijze verbeteren?"

Op basis van een vooronderzoek in de vorm van een literatuurstudie naar het begrip kwaliteit en de beheersing van een software lifecycle in een vergelijkbare markt is er een definitie vastgesteld waarmee de kwaliteit van een software lifecycle wordt benaderd. Op basis van die benadering en de opgedane kennis van het vooronderzoek zijn er interviews afgelegd om de huidige software lifecycle van Thinkwise in kaart te brengen. Om dit te toetsen op kwaliteit is er een literatuurstudie uitgevoerd om de best-practices in kaart te brengen. Aan de hand van dat literatuuronderzoek is er een GAP-analyse uitgevoerd om de verschillen in kaart te brengen tussen de best-practice en de huidige situatie. Op basis van de resultaten van die GAP-analyse zijn er procesmatige verbeteringen en alignmentverbeteringen opgesteld en is er een focusgroep georganiseerd om de alignmentverbeteringen in de praktijk te plaatsen door metingen op te stellen en kritische processen te identificeren. De opgestelde metingen en processen zijn vervolgens weer getoetst door middel van een literatuurstudie. Ten slotte is er een technische analyse gedaan door middel van interviews om de implementatiemogelijkheden te onderzoeken om zo een stappenplan te realiseren waarmee de oplossingen in de organisatie kunnen worden geïmplementeerd.

Het ontbreken van het inzicht in de kwaliteit van het product is een gevolg van een knelpunt in de alignment van de organisatie. Het belangrijkste knelpunt in deze alignment blijkt in de relatie tussen de uitvoering van de software lifecycle en het kwaliteitsmanagement in een project. Er ontbreekt een constante evaluatie na de uitvoering van de ontwikkelprocessen waardoor er geen tussentijds inzicht is en dit de uitvoering van kwaliteitsmanagement belemmerd. Het vergroten van de kwaliteit van de software lifecycle zal mogelijk worden door na elke fase een evaluatie uit te voeren door de processen in de fase meetbaar te maken in de SF waardoor kwaliteitsmanagement in een project mogelijk wordt.

De volgende aanbevelingen zijn gemaakt op basis van de resultaten:

- Een dashboard ontwikkelen met het opgestelde stappenplan waarin er metingen worden gedaan op basis van elke fase in de software lifecycle. De resultaten gebruiken als input voor kwaliteitsmanagement in een project en het ontwikkelde dashboard opnemen in een nieuwe versie van de Thinkwise projectmethodiek.
- Projectdocumentatie opnemen in de SF om zo automatische evaluaties mogelijk te maken.
- Projectverantwoordelijken betrekken bij het ontwikkelen van de Thinkwise projectmethodiek om weerstand tegen veranderingen te voorkomen en te zorgen voor gedeelde eigenaarschap.
- Onderzoeken of een beheermodel als ASL kan zorgen voor meer inzicht in eventuele alignment-problemen om hier zo een betere beheersing op te krijgen.

Voorwoord

Met het inleveren van dit onderzoek sluit ik mijn bachelor Business IT & Management af. Vier jaar geleden startte ik deze opleiding nadat ik was toegelaten door middel van een 21+ toets. Het is de grootste uitdaging van mijn leven geweest en ik had vier jaar geleden nooit gedacht dat ik dit met een succes af zou ronden, laat staan zonder enige vertraging. Nu ligt het voor uw neus; mijn scriptie. Ik ben bijzonder trots op mijn prestatie. Deze opleiding heeft mijn leven veranderd.

De afstudeerperiode was een stressvolle tijd. Desondanks heb ik veel vrijheid gekregen van Thinkwise om het onderzoek te volbrengen en tot een succes af te ronden. Thinkwise heeft daarbij altijd de middelen gegeven om dit onderzoek te kunnen doen en alles was mogelijk. Dit maakt het een uitstekende opdrachtgever.

Als eerste wil ik Hans Schokker bedanken voor de coaching tijdens de uitvoering van dit onderzoek. Het was prettig om soms even te sparren over bepaalde richtingen en het was fijn dat er altijd een plek was om even te skypen of langs te komen in Deventer.

Dan wil ik graag Mark van den Berg bedanken voor het mogelijk maken van dit onderzoek binnen deze kaders en het meedenken met de oplossingen. De vrijheid en mogelijkheid die mij is gegeven om dit onderzoek te volbrengen in de laatste stadia van de periode heeft ervoor gezorgd dat ik het tot een succes kon afronden. Ook wil ik graag Moller Toma en Bram Vervloet bedanken voor het delen van hun kennis en het meedenken in een oplossingsrichting. De kickstart gaf mij een goede start aan het begin van dit traject.

Tevens wil ik Robert van Tongeren, Harold Soepenberg, Robert-Jan de Nie, Gert Groenenveld en Esther Kranendonk bedanken voor het mogelijk maken van hun tijd, het delen van hun kennis en het meedenken met het in kaart brengen van de Software Lifecycle.

Als laatste, en voor mij het belangrijkste, wil ik graag mijn vriendin Emma van der Tang bedanken. In deze hectische periode heb ik pieken en dalen gekend. Jij hebt mij hier doorheen weten te loodsen. Van middagen typen, het aanhoren van mijn geklaag tot aan het meedenken met zaken waar je niks van snapt. Zonder jou was ik niet waar ik ben.

Ik wens u veel leesplezier!

Davey Brom
15 juni 2019
Apeldoorn

INHOUDSOPGAVE

1. INLEIDING

1.1. Aanleiding

Dit onderzoek heeft plaatsgevonden bij Thinkwise Software B.V. Al 17 jaar ontwikkelt Thinkwise Software haar eigen Low Code platform. Dit platform is vooral ontwikkeld om klanten een duurzame IT-oplossing te bieden en hen “Future-proof” te maken.
Thinkwise heeft dan ook de missie:
“Zorgen voor alternatieve bedrijfssoftware waar bedrijven wél blij van worden.”
Thinkwise verstaat onder “blij worden”:

- Software naar je hand kunnen zetten
- Alleen die software gebruiken die je nodig hebt
- Eenvoudig kunnen aanpassen tijdens het ‘maakproces’
- Software die niet veroudert
- De beste businesscase en ROI hebben
- Gebruikers blij maken met onze software (Thinkwise Software B.V., 2019).

Volgens Mark van den Berg (CSO) is Thinkwise de laatste jaren, vooral door het aantrekken van grote klanten, sterk gegroeid. Dat wil ook zeggen dat er steeds grotere projecten worden uitgevoerd. Bij de uitvoering van de recentere ‘grotere’ projecten hebben veel projectleiders gerealiseerd dat er weinig inzicht is in de kwaliteit van het project en het product. Van den Berg stelt dat er op dit moment pas inzicht is in de kwaliteit van het product op het moment dat het daadwerkelijk opgeleverd wordt en wordt getest. Dit moet volgens hem duurzaam verbeterd worden, zodat er kwaliteitscontroles in het proces worden opgenomen en er inzicht komt in het ontwikkelproces. Om dit zo volledig mogelijk te onderzoeken wordt dit onderzoek gericht op de volledige software lifecycle van Thinkwise. Dit met de gedachte dat een beter inzicht in de lifecycle leidt tot een betere kwaliteit lifecycle en dus onder aan de streep tot een betere kwaliteit van het eindproduct.

2. ONDERZOEKSOPZET

In dit hoofdstuk wordt het probleem geschetst. De probleemstelling dient als uitgangspunt voor het onderzoek. Vervolgens worden de vragen geformuleerd die in dit onderzoek behandeld worden.

2.1. Probleemstelling

In de afgelopen jaren geniet Thinkwise van een uitstekende groei. Grote opdrachtgevers worden aangetrokken en daarbij behorende grote projecten worden intern gestart om die nieuwe klanten te voorzien van een product. De projecten van de afgelopen tijd worden ook constant geëvalueerd. Uit deze evaluatie is vastgesteld dat er te weinig inzicht is in de kwaliteit van het product en de processen in het project. Volgens Mark van den Berg wordt de kwaliteit van het product in de huidige situatie in veel gevallen pas echt helder als het product klaar is en bij de livegang van de gerealiseerde software komen managers bij veel projecten geregeld voor verrassingen te staan. Dit kunnen bugs zijn of andere afwijkingen waar de klant niet blij mee is. Thinkwise heeft klanttevredenheid hoog in het vaandel staan en wil hier dus verandering in aanbrengen..

2.2. Doelstelling

Het onderzoek richt zich op de volgende doelstelling:

- De kwaliteit van de software lifecycle van Thinkwise op een duurzame wijze verbeteren.

Om deze doelstelling te bereiken zijn ook de volgende doelstellingen nodig:

- Een beeld schetsen van de huidige software lifecycle.
- Inzicht krijgen in de kwaliteit van de huidige lifecycle.
- Het in kaart brengen van de veranderingen die noodzakelijk zijn voor de verbeteringen van de kwaliteit van de lifecycle.
- Het verschil tussen de huidige en de gewenste situatie in kaart te brengen wat als basis dient voor de veranderstrategie.
- Een tool ontwikkelen om inzicht in de kwaliteit van de lifecycle te vergroten.

2.3. Onderzoeksvragen

Om de bovenstaande doelstellingen en eindresultaat te behalen is er besloten om een onderzoek te doen naar de kwaliteitsinzicht in de software lifecycle binnen Thinkwise. Voor dit onderzoek is de volgende hoofdvraag bedacht:

“Hoe kan Thinkwise de kwaliteit van haar software lifecycle op een duurzame wijze verbeteren?”

De hoofdvraag is opgesteld om de belangrijkste doelstelling te bereiken. Het is dan ook de hoe-vraag op de doelstelling.

De hoofdvraag wordt beantwoord door middel van de volgende deelvragen en sub-deelvragen:

- Wat wordt er verstaan onder kwaliteit en hoe wordt dit beheerst in een organisatie?
  - Wat wordt er verstaan onder kwaliteit volgens de literatuur?
  - Wat is kwaliteit volgens Thinkwise?
  - Hoe wordt kwaliteitsmanagement toegepast in de IT?
  - Hoe wordt kwaliteitsmanagement toegepast bij Thinkwise?
- Wat maakt een software lifecycle succesvol?
  - Wat is een software lifecycle?
  - Wat is de best-practice op het gebied van software lifecycles in de IT?

        - Wat maakt de benadering van deze best-practices succesvol?
- Wat is de huidige kwaliteit van de gehanteerde Software Lifecycle?
    - Welke processen vallen onder de software lifecycle van Thinkwise?
    - Welke processen worden, volgens de literatuur, goed gehanteerd?
    - In welke mate ondersteund de huidige alignment de software lifecycle?
- Hoe kan de kwaliteit van de software lifecycle verbeterd worden?
    - Wat is er nodig om tot de gewenste lifecycle te komen?
        - Wat is er nodig op procesmatig gebied?
        - Wat is er nodig op IT-alignment gebied?
    - Hoe kan een dashboard bijdragen aan de gewenste lifecycle?
- Hoe moeten de benodigde verbeteringen worden geïmplementeerd?
    - Wat is er op procesmatig gebied nodig om de gewenste verbeteringen te implementeren?
    - Wat is er op alignment gebied nodig om de gewenste verbeteringen te implementeren?

3. THEORETISCH KADER

Voor het beantwoorden van de deelvragen is het in sommige gevallen nodig om een verduidelijking te geven van de gebruikte literatuur. Omdat er in dit onderzoek veel vergelijkingen worden gedaan, is ervoor gekozen om ook de benodigde informatie te verschaffen die de vergelijking mogelijk hebben gemaakt. De verschillende vergaarde informatie wordt vervolgens in hoofdstuk 5 gebruikt voor vergelijking of onderbouwing. In dit theoretisch kader wordt er eerst een achtergrond gegeven over kwaliteit en kwaliteitsmanagement. Vervolgens worden verschillende software lifecycles en lifecycle-methodes uit de praktijk in kaart gebracht. Dan wordt er een beeld geschetst van de beheermodellen en de daarbij onderbouwde verdieping naar ASL en ten slotte een korte beschrijving van het WWK-model.

3.1. Kwaliteit

Om te spreken over kwaliteit is het belangrijk om eerst een helder beeld te krijgen wat kwaliteit definieert. Kwaliteit is een relatief begrip. Het wordt vaak omschreven als een individuele mening van “de uitmuntendheid van een object of een bepaald voorwerp”. Als er vervolgens de vraag wordt gesteld wat Kwaliteit dan precies inhoudt heeft men daar vaak geen antwoord op. (Trienekens J. J., 1994). Om de vergelijking zo goed mogelijk te kunnen doen en uiteindelijk een definitie te bepalen is er gekozen om een triangulatie te doen van de literatuur. Er wordt veel geschreven over kwaliteitsdefinities, maar het grootste deel van de literatuur verwijst naar de volgende drie partijen: Garvin, Juran en ISO-9001. Deze bronnen worden dan ook gebruikt voor de triangulatie.

3.1.1. Kwaliteit volgens Garvin

David A. Garvin heeft in 1984 een onderzoek gedaan naar productkwaliteit in de productie-industrie. Hier zijn de volgende verschillende soorten kwaliteitsdefinities opgesteld.

Transcendent-gebaseerd

Volgens de transcendent-gebaseerde benadering staat kwaliteit gelijk aan “aangeboren excellentie”. Het is zowel absoluut als universeel herkenbaar, een teken van een compromisloze standaard en het behalen van een grote prestatie.

Gebruiker-gebaseerd

Deze definitie beschouwd kwaliteit als de mate waarin een product of voorwerp voldoet aan de eisen van de gebruiker. Hierbij is kwaliteit een subjectief begrip. Het is de mate waarin verschillende individuele wensen en behoefte van een gebruiker worden gecombineerd tot een product wat volgens de gebruikers dan kwaliteit betekent. Het is daarbij de vraag of het product, in dat geval, echt kwaliteit bevat of het ‘slechts’ aan de eisen voldoet.

Product-gebaseerd

De product-gebaseerde definitie stelt dat kwaliteit wordt bepaald door vooraf eenduidige kenmerken. De objectieve kenmerken kunnen worden gemeten en getoetst. Ze worden vastgesteld door de normen die gelden binnen de gestelde discipline. . Kwaliteit moet hierin ten alle tijden kwantitatief toetsbaar zijn.

Fabricage-gebaseerd

Deze definitie betreft de kwaliteit die betrekking heeft op de mate waarin producten voldoen aan de vooraf gestelde requirements. Door tijdens het ontwikkelproces de (deel)producten te meten, kan kwaliteit worden vastgesteld. Volgens deze definitie kan de kwaliteit bewerkstelligt worden door in het productieproces te focussen op het oplossen en signaleren van afwijkingen en fouten in het proces zelf.

Waarde-gebaseerd

Deze definitie stelt dat kwaliteit geen absoluut gegeven is, maar dat het moet worden verhouden met tijd, kosten en inspanning. Ten opzichte van de andere definities wordt er in deze definitie het maken van afwegingen sterk benadrukt. De afwegingen hebben vaak betrekking op de mate waarin een product voldoet aan kwaliteit

Elke definitie heeft dus een andere inhoud en scope. Het zit hier vooral in de mate van absoluutheid en de mate van objectiviteit. Waar kwaliteit volgens de waarde-gebaseerde definitie nooit een absoluut gegeven is, het dit het juist bij de product-gebaseerde definitie wel; “De optelsom van een verzameling objectieve kwaliteitskenmerken”. (Garvin, 1984) (Trienekens J. , 1994)

3.1.2. Kwaliteit volgens Juran

Volgens Joseph M. Juran, de vader van het hedendaags kwaliteitsmanagement en schrijver van het boek; *Juran’s Quality Handbook*, zijn er twee kritische benaderingen van het woord kwaliteit:

1. Kwaliteit betekent de set van eigenschappen die voldoen aan de behoefte en wens van de klant en zorgen voor klanttevredenheid. In deze benadering is kwaliteit gericht op inkomen. Het doel van een hogere kwaliteit is het behalen van een hogere klanttevredenheid en het verhogen van de omzet. Echter is het, om hogere kwaliteit te behalen, nodig om hierin te investeren. Een hogere kwaliteit staat dus gelijk aan hogere kosten.
2. Kwaliteit betekent *Vrijheid van afwijkingen*-vrijheid van fouten die ervoor zorgen dat werk opnieuw gedaan moet worden of die zorgen voor praktijkfouten, klant-ontevredenheid, klachten etc. In deze benadering is kwaliteit gericht op kosten en kost kwaliteit geregeld minder.

Product features that meet customer needs	**Freedom from deficiencies**
Higher quality enables companies to: Increase customer satisfaction Make products salable Meet competition Increase market share Provide sales income Secure premium prices The major effect is on sales. Usually, higher quality costs more.	Higher quality enables companies to: Reduce error rates Reduce rework, waste Reduce field failures, warranty charges Reduce customer dissatisfaction Reduce inspection, test Shorten time to put new products on the market Increase yields, capacity Improve delivery performance Major effect is on costs. Usually, higher quality costs less.

FIGUUR 1: VERSCHILLENDE KWALITEITSBENADERINGEN VOLGENS JURAN

Figuur 1 helpt ons om te begrijpen waarom verschillende betekenissen van kwaliteit eindigen in verwarring. (Juran, 1998)

3.1.3. Kwaliteit volgens ISO 9001

Om de definitie van kwaliteit in de kaders te plaatsen, is het goed om ook de definitie van kwaliteit volgens de ISO 9001 norm mee te nemen. ISO 9001 wordt uitgelegd in hoofdstuk 3.2.4.

“*Kwaliteit is de mate waarin een product of dienst voldoet aan de normen, behoefte en/of specificaties in overeenkomst met de klant*”. (Institute of Technology Tallaght, 2007) (Stichting Nederlands Normalisatie-Instituut , 1994) ISO-9001 is een deel van het normalisatie-instituut die zich vooral bezig houdt met kwaliteitsmanagement en kwaliteitssystemen.

3.2. Kwaliteitsmanagement-methoden

Om kwaliteitsmanagement-methodes in kaart te brengen vragen we ons eerst de vraag wat er wordt verstaan onder kwaliteitsmanagement. Vervolgens plaatsen we deze methodes in het perspectief van het onderzoek. Er wordt onderzocht hoe kwaliteit wordt gewaarborgd. Het management van kwaliteit wordt in kaart gebracht en belangrijke definities zoals; kwaliteitsmanagement en een kwaliteitsmanagementsysteem worden gegeven. We richten ons op gepaste methoden in de IT-sector.

3.2.1. Kwaliteitsmanagement

ISO 9001, die zich vooral bezig houdt met kwaliteitsmanagement en kwaliteitssystemen, houdt de volgende definitie van kwaliteitsmanagement aan:

"*Alle activiteiten binnen het totale management-orgaan die die kwaliteitseisen, doelen en verantwoordelijkheden bepalen en deze implementeren door kwaliteits-planning, -controle en -verbeteringen toe te passen*"

Kwaliteitsmanagement is de verantwoordelijkheid van alle managementlagen en moet worden gestuurd door het topmanagement. Elke medewerker is betrokken bij de implementatie hiervan. Economische aspecten worden vaak meegenomen bij deze vorm van management. (Stichting Nederlands Normalisatie-Instituut , 1994)

3.2.2. Kwaliteitsmanagementsysteem

Naast kwaliteitsmanagement te definiëren, geeft ISO 9001 ook een definitie voor een kwaliteitsmanagementsysteem. Een praktische omschrijving:

"Een doelmatige manier van werken binnen een organisatie die er toe leidt dat de organisatie goede producten en diensten levert waar de klanten tevreden over zijn." (Stichting Nederlands Normalisatie-Instituut , 1994)

In veel gevallen voldoen kwaliteitssystemen niet aan de verwachtingen. Dit kan liggen aan het ontwerp en/of de implementatie van het kwaliteitssysteem. Wat ook een veel voorkomende reden is zijn verkeerde en overtrokken normen. Om aan deze normen te voldoen worden er in de praktijk vooral managementmethoden gebruikt die zijn toegespitst op deze normen. (Zijlstra, 2010)

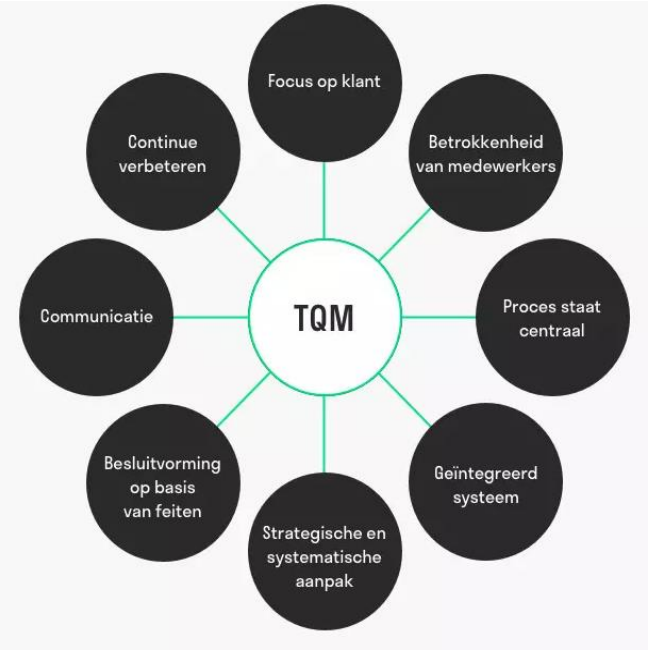
3.2.3. Total Quality Management

Een voorbeeld van een kwaliteitsmanagementmethode is TQM. Dit kan in de kaders van het onderzoek worden geplaatst omdat na onderzoek geconcludeerd is, dat de TQM aanpak een goede oplossing is voor kwaliteitsverbeteringen in de IT. (Parzinger & Nath, 2000) (Zhang, 1997)

Total Quality Management (TQM) is een gestructureerde managementaanpak waarbij de focus ligt op continue verbetering van producten en diensten op basis van feedback. De aanpak is, sinds zijn ontwikkeling in 1954, doorontwikkeld tot een concept die in elke sector kan worden ingezet.
Het doel van TQM is het 'in één keer goed doen'. Hierdoor bespaart de organisatie tijd op correcties, fouten en mislukte service- en productimplementatie.
In figuur 2 bevat een schematische weergave van TQM. Het laat de verschillende facetten van TQM zien die uiteindelijk zorgen voor een totale kwaliteitscontrole.

Het gaat om het combineren van de volgende principes:

- Klantgerichtheid
- Betrokken medewerkers
- Processen centraal in de organisatie
- Geïntegreerd systeem (bv. ISO 9000)



FIGUUR 2: MODEL TOTAL QUALITY MANAGEMENT

- Strategische en systematische benadering
- Beslissen op basis van feiten, niet op meningen
- Communicatie op basis van strategie
- Continue verbetering

Het implementeren van TQM is niet zomaar een model. Het is een gedachtegoed die ingebakken moet worden in de organisatie en de cultuur die er heerst. (Hackman & Wageman, 1995) (van Vliet, 2019) TQM kan zowel afzonderlijk worden ingericht als een te volgen normenset als de ISO 9000-serie.

3.2.4. De ISO 9000-serie

De ISO 9000-serie is een verzameling van standaarden van het ISO (International Organisation for Standardization) instituut die vastleggen hoe een organisatie haar kwaliteit kan waarborgen. Aantoonbare klanttevredenheid en continue verbeteringen zijn hierbij essentiële onderwerpen. Door effectieve toepassing van het systeem, uitvoering van processen die zijn gericht op verbetering en het voorkomen van non-conformiteit, wordt voor (aantoonbare) klanttevredenheid gezorgd.

De ISO 9000-serie bestaat uit de volgende standaarden:

- ISO 9000:2005 – Kwaliteitsmanagementsystemen - Grondbeginselen en definities
- ISO 9001:2015 – Kwaliteitsmanagementsystemen- Eisen
- ISO 9004:2018 – Richtlijnen voor prestatieverbeteringen
- ISO 19011:2018 – Richtlijnen voor het uitvoeren van kwaliteits-/millieumanagementsaudits

Het gaat hier dus niet om een methode of model maar om specifieke eisen/normen binnen een organisatie. (Marquardt, 1998)

3.3. De Software Lifecycle

Dit onderzoek gaat over het kwalitatief verbeteren van de software lifecycle. Het is daarom, naast het onderzoeken van het begrip kwaliteit, goed om tevens een beeld te geven van wat een software lifecycle inhoudt. De invulling van een software lifecycle is sterk afhankelijk van de gehanteerde processen in een organisatie. Er zijn echter veel processen met elkaar te combineren en onder te brengen in een aantal categorieën. Om een duidelijk beeld te krijgen van verschillende Software Lifecycles in de praktijk worden er twee verschillende uitwerkingen onderzocht.

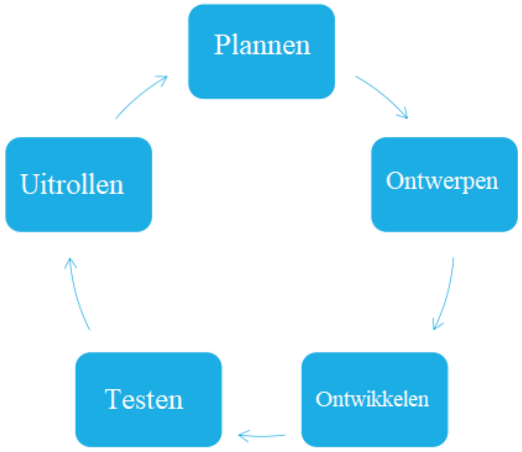
3.3.1. Definitie Software Lifecycle

De Software Lifecycle, ook wel Software Development Life Cycle (verkort: SDLC) is een workflow proces welke de kernfasen en activiteiten in een ontwikkelcyclus definiëren (Wong Tiky, 2016). Het is gericht op het ontwikkelen van software met de hoogst mogelijke kwaliteit en de laagste kosten in de kortste tijd. Het omvat een gedetailleerd plan voor het ontwikkelen, aanpassen, onderhouden en vervangen van een informatiesysteem. (Stackify, 2019)

3.3.2. Software lifecycle in de praktijk

Software Lifecycle volgens Wong Tiky

De activiteiten kunnen worden gegroepeerd in vijf categorieën: Plannen, Ontwerpen, Ontwikkelen, Testen, Uitrollen.



FIGUUR 3: SOFTWARE LIFECYCLE VOLGENS WONG TIKY

Plannen

- Scope bepalen
- Haalbaarheid onderzoeken
- Kwaliteits- en risicomanagement

Design

- Requirements opstellen
- Ontwerp maken
- Ontwerpen beoordelen

Ontwikkelen

- Software realiseren
- Functionele requirements opstellen

Testen

- Test uitvoeren door programmeurs
- Test uitvoeren door eindgebruikers
- Test uitvoeren door kwaliteitsexperts
- Testplannen ontwikkelen

Uitrollen

- Systeem controlleren op correctheid
- Implementatieplan opstellen
- Deploymentdocumentatie ontwikkelen (installatie-, administratie- en gebruikershandleiding) (Wong Tiky, 2016)

3.3.3. Software Development Life Cycle model

Het model in Figuur 4 is een goede weergave van de verschillende fasen in een software lifecycle. Het model bevat de meest voorkomende fasen in veel verschillende lifecycles in de praktijk. Elke fase produceert deelproducten die weer nodig zijn voor volgende fasen in de lifecycle. In deze vorm komen de volgende fasen en activiteiten aan bod:

Planning (Plannen)

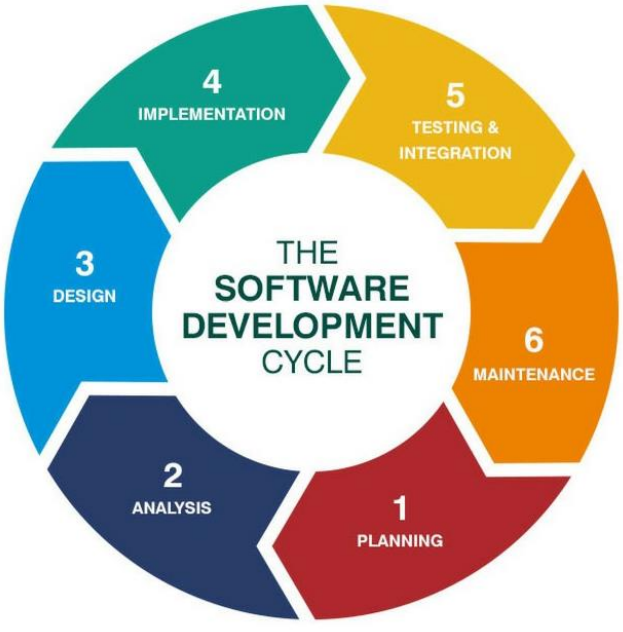
- Identificatie van het systeem voor ontwikkeling
- Haalbaarheidstest
- Creatie van het projectplan (Pinheiro, 2018)

Analysis (Analyse)

- Verzamelen van business-requirements
- Procesdiagrammen opstellen
- Een gedetailleerde analyse uitvoeren (Imam, 2018)

Design (Ontwerp)

- Ontwerp van de IT infrastructuur



FIGUUR 4: HET SOFTWARE DEVELOPMENT LIFECYCLE MODEL

- Ontwerp van het systeemmodel (Pinheiro, 2018)

Implementation (Implementatie)

- Software ontwikkelen

Testing & Integration (Testen & Integratie)

- Schrijven van testcases
- Uitvoeren van testcases (Imam, 2018)

Maintenance (Onderhoud)

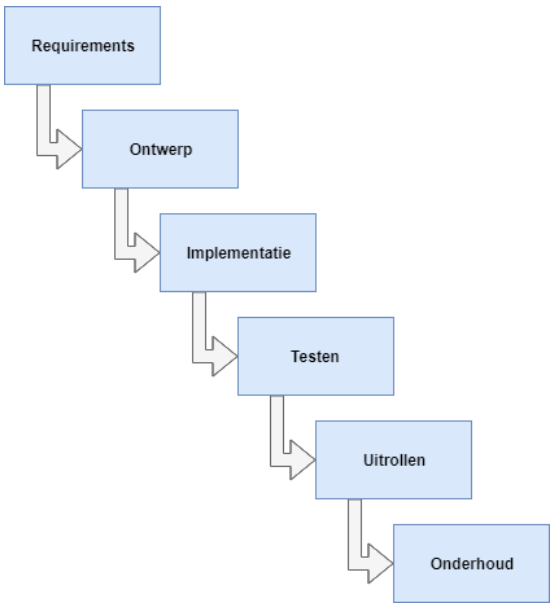
- Support van gebruikers
- Systeemonderhoud
- Systeemaanpassingen en -veranderingen. (Imam, 2018)

3.4. Software Development Lifecyclemethoden

Om een succesvolle software lifecycle te beheersen kunnen er verschillende methoden worden ingezet. Deze fungeren als methodieken die van zichzelf een lifecycle creëren. Elk model doorloopt een serie van unieke stappen die moeten aansluiten bij de vorm van een gekozen project en invulling van ontwikkelprocessen om zo de kwaliteit van de hantering en de lifecycle zelf te garanderen. De modellen zijn onder elkaar gezet en onderzocht op toepasbaarheid en de voor- en nadelen.

3.4.1. Watervalmethode

De watervalmethode is een van de eerste, maar meest bekende en gebruikte methodologie. Het is een gemakkelijk te begrijpen en te gebruiken lifecycle. Elke fase moet compleet worden afgerond voordat de volgende fase kan worden voortgezet. De output van een fase dient als input voor de volgende (Wong Tiky, 2016). Documentatie en testen worden aan het eind van een fase pas geïnitieerd. Omdat niet altijd hetzelfde team aan het gehele proces werkt, en er ten allen tijden 1 fase per keer wordt uitgevoerd, is deze methode vaak tijdrovend en kost het relatief veel geld. In de watervalmethode worden defecten pas laat in de lifecycle gevonden omdat het testteam niet wordt betrokken bij het begin van het project. (Balaji & Sundararajan Murugaiyan, 2012).



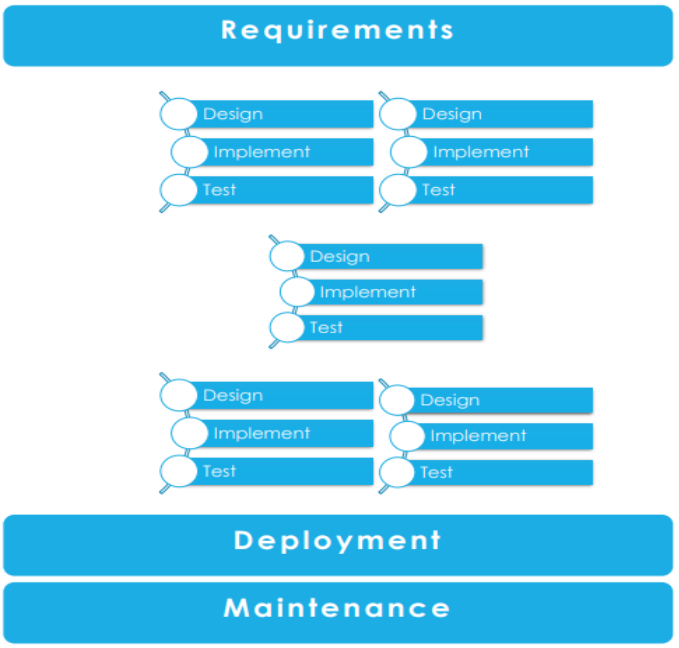
FIGUUR 5: WATERVALMETHODE

Toepasbaarheid:

- Weinig tot geen veranderingen of niet-goedgekeurde requirements
- Software dat goed gedocumenteerde documenten nodig heeft
- Gebruik van volwassen en minder dynamische technologieën
- Managers hebben genoeg middelen en experts om haar rol te grijpen bij elke fase
- Simpele en kleine projecten

3.4.2. Iteratieve methode

Het iteratieve model werkt puur op vereenvoudigde requirements. Dit zijn een subset van de software of de applicatie-requirements. Het product wordt iteratief verbeterd en geëvalueerd naar het uiteindelijke product voor inzet. Elke iteratie wordt dus apart gebouwd en bevat de volgende fasen: Design, implementatie en test. Bij het iteratieve model wordt de software dus ontwikkeld in kleine blokken. Dit wordt echter aan het eind van het project gezamenlijk uitgerold en opgeleverd. (Pinheiro, 2018) (Balaji & Sundararajan Murugaiyan, 2012)



FIGUUR 6: ITERATIEVE METHODE

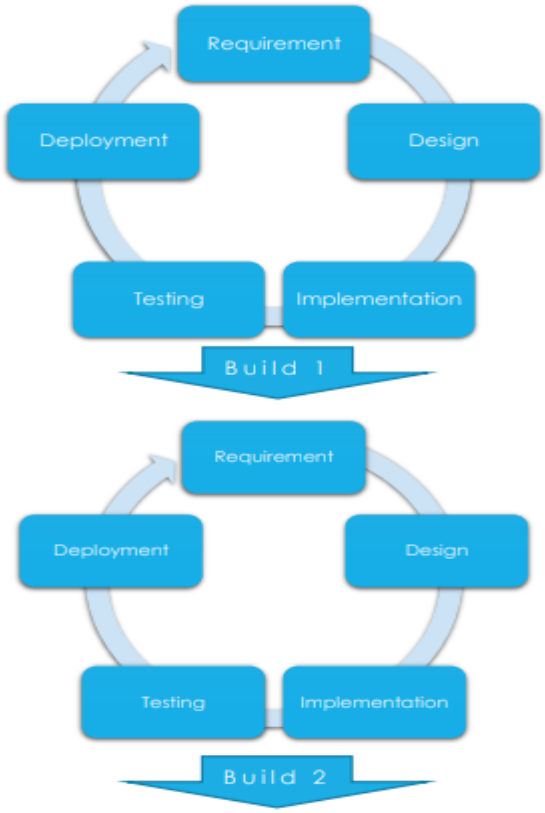
Toepasbaarheid

De toepasbaarheid van de iteratieve methode is in de volgende situaties het beste:

- Grote requirements zijn gedefinieerd maar de belangrijkste details kunnen veranderen over tijd.
- Nieuwe technologieën worden gebruikt en er is een leercurve voor de programmeurs.
- Resources zijn te gelimiteerd voor een groot project, dus worden er kleinere projecten gedraaid. Hierdoor zijn teamgenoten meer in contact.
- Een groot projectrisico omdat het doel van het project kan veranderen over tijd.

3.4.3. Agile methode

De agile methode vergroot de voordelen van het iteratieve model en richt zich op klanttevredenheid, productaanpasbaarheid en de snelle oplevering van het product. Opgedeeld in de fasen van requirements tot het uitrollen breekt de agile methode de software op in delen. In plaats van terug te gaan naar de ontwerpfase, zoals de iteratieve methode dat doet na elke subset te testen, vervolgt de agile methode zich met het uitrollen van de software en implementeert dit product. Klanttevredenheid wordt hier behaald door snel producten op te leveren van kleine stukken behulpbare software. (Balaji & Sundararajan Murugaiyan, 2012) (Wong Tiky, 2016)



FIGUUR 7: AGILE METHODE

Toepasbaarheid

De toepasbaarheid van de agile methode is in de volgende situaties het beste:

- Gedetailleerde informatie vanuit de business users is niet nodig om te starten
- Project wordt gedreven door functionaliteit
- Product requirements veranderd dynamisch
- Resources op testen.
- Nauwe samenwerking binnen het ontwikkelteam
- Nauwe relatie met eindgebruikers

3.4.4. Outputdocumenten Software Lifecycles

Tijdens de uitvoering van de bovenstaande best practices worden er verschillende outputdocumenten gegenereerd. Deze documenten worden in elke methodiek benoemd als de output van een fase. Om een beter beeld te krijgen van wat deze documenten inhoudt wordt er een beschrijving van de documenten gegeven:

Business Case documentatie
Output van fase: Requirements
Dit document wordt opgesteld door projectmanagers. Het beschrijft de omgeving en de organisatie van de klant. Alle details van de klantbehoefte worden zoveel mogelijk genoteerd en de scope van het project wordt bepaald. Vervolgens worden de middelen en benodigdheden gedefinieerd.

Software Requirements
Output van fase: Ontwerp
Vanuit de analyse van de business worden zowel de functionele- als de gebruikersrequirements opgesteld. Het geeft een beeld van de klantbehoefte en de eisen waar het systeem aan moet voldoen.

Ontwerpspecificatie
Output van fase: Ontwerp
Aan de hand van de requirements worden de ontwerpspecificaties opgesteld. Het is een omschrijving van hoe het systeem eruit komt te zien. Dit begint met een high-level ontwerp. Uiteindelijk worden er duidelijke ontwerpspecificaties opgesteld waarbij de hardware en de software worden meegenomen in het ontwerp hiervan.

Functionele specificaties
Output van fase: Implementatie
Als ontwikkelaars aan de hand van de bovenstaande documenten de software realiseren, worden alle details van de ontwikkelde functionaliteiten gedocumenteerd in functionele specificaties.

Testplannen
Output van fase: Testen
Alle ontwikkelde functionaliteiten die zijn ontwikkeld in de implementatiefase worden opgenomen in het testplan om de werking hiervan te testen in de praktijk. Een testplan bevat een checklist om te checken of een functionaliteit naar behoren werkt. Tevens wordt er getest of de software voldoet aan de gebruikersrequirements.

Deploymentplan
Output van fase: Uitrollen
In het deploymentplan wordt uitgelegd hoe een opgeleverd stuk software kan worden uitgerold in de productieomgeving.

Calamiteitenplan
Output van fase: Uitrollen
Het calamiteitenplan wordt opgesteld om te documenteren waar de risico's in de applicatie zich bevinden en hoe dit in de toekomst kan worden verkleind. Tevens wordt er gedocumenteerd wat te doen als er een calamiteit bij dat risico wordt voorgedaan. Dit document dient als basis voor de onderhoudsfase. (Wong Tiky, 2016) (Balaji & Sundararajan Murugaiyan, 2012)

3.5. Beheermodellen

Een organisatie heeft tegenwoordig een grote uitdaging aan het beheer van haar informatiesystemen. Een toepassing, methode of hantering van een software lifecycle valt ook in een informatiesysteem van de organisatie. Het is daarom goed om ook beheermodellen onder de loep te nemen. Om dit zo goed mogelijk te beschrijven wordt eerst het doel beschreven, vervolgens de beheertypen en ten slotte de toepassing van een beheermodel en de daarbij toepasbare best-practice.

3.5.1. Doel beheermodel

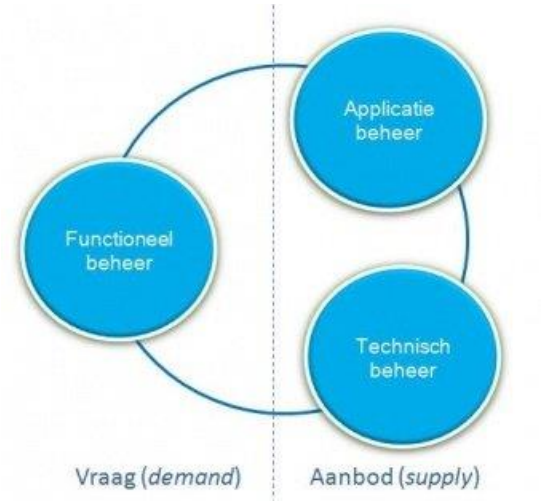
Het is allereerst goed om te onderzoeken waarom een beheermodel is ontwikkeld. Looijen stelt dat het doel van een beheermodel is; *"Een efficiënt en effectief managementsysteem te bieden voor het beheer van informatiesystemen"* (Looijen & Delen, 1992)

3.5.2. Beheertypen

In het boek van Looijen worden er in essentie drie deelgebieden van IT beheer onderkend; Functioneel-, technisch- en applicatiebeheer. De reden van de deelgebieden is omdat men in het beheer clusters van taakgebieden aantreft die wat betreft inhoud, verantwoordelijkheid, kennis en hulpmiddelen principieel van elkaar verschillen. (Looijen & Delen, 1992) De driedeling legt ook de basis voor de functiescheiding tussen vraag (demand) en aanbod (supply), een van de meest belangrijke governance-principes. Een visueel beeld van de scheiding is te vinden in Figuur 8: Beheermodel van Looijen.

Functioneel beheer

Het functioneel beheer is een beheervorm die alle beheertaken omvat die nodig zijn voor het gebruik van informatiesystemen. Aangezien gebruik is gericht op functionaliteiten zoals; invoeren, manipuleren, verkrijgen, transporteren en opslaan van gegevens, wordt gesproken over functioneel beheer. Het beheermodel van Looijen geeft ook duidelijk aan dat functioneel beheer geen ICT-functie is, maar een business-functie (vraag-zijde). (Looijen & Delen, 1992) (Hoving & van Bon, 2013) Functioneel beheer fungeert dus eigenlijk als eigenaar en opdrachtgever voor het informatiesysteem. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)



FIGUUR 8: BEHEERMODEL VAN LOOIJEN

Applicatiebeheer

Applicatiebeheer betreft het ontwikkelen en onderhouden van applicaties in een organisatie. Het is een beheervorm die over het taakgebied applicatie-onderhoud heen is gebouwd. Het gaat hier om het beheren van de continuïteit van de organisatie met betrekking tot het wijzigen van de applicatieprogrammatuur en de applicatieonderhoud. (Looijen & Delen, 1992) Het gaat hier dus om de partij die het informatiesysteem (applicatie) ontwikkeld, beheert en/of onderhoudt. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Technisch beheer

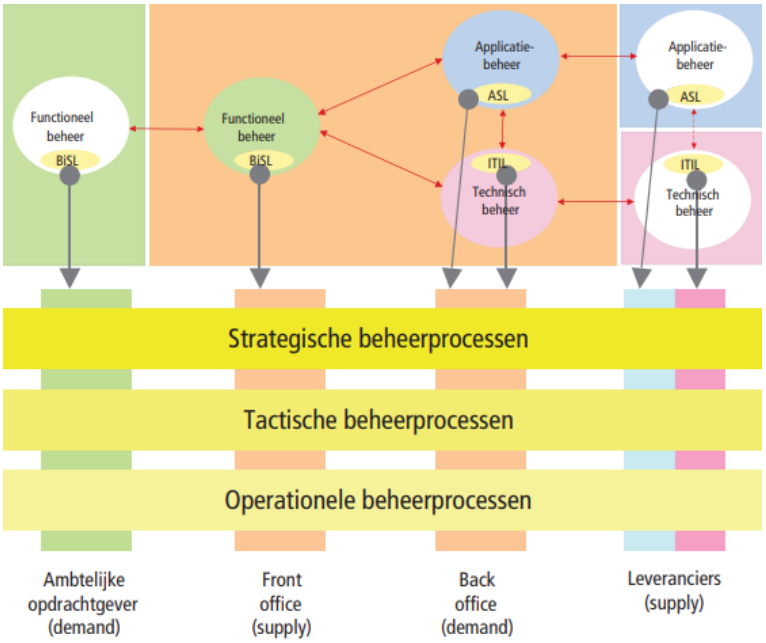
Als hekkensluiter is er nog het technisch beheer. Dit omvat als beheervorm alle taken die nodig zijn voor het installeren, accepteren en operationeel maken en houden van informatiesystemen en technische infrastructuren. Hierbij wordt ook het optimaliseren van verwerkingsprocessen en het aanbrengen van wijzigingen in de technische infrastructuur meegenomen. (Looijen & Delen, 1992) (Hoving & van Bon, 2013) Het is kortweg de organisatie die de informatiesystemen draait en zorgt

dat de infrastructuur op orde blijft. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

3.5.3. Toepassing Beheermodellen

Met de kennis van bovenstaande functiedelingen zijn er in de laatste jaren veel verbeter- en professionaliseringsslagen gemaakt. Zeker omdat Looijen stelt dat de verschillende beheergebieden optimaal samen moeten werken om de beste kwaliteit van beheer te verstrekken. (Looijen & Delen, 1992)

Vanuit deze gedachtegang zijn er in de afgelopen jaren verschillende best-practices ontwikkeld die opereren op een bepaald beheergebied. BiSL voor Functioneel Beheer, ASL voor Applicatiebeheer en ITIL voor Technisch beheer. Zie Figuur 9: Schematische weergave beheermodellen voor een schematische weergave.



FIGUUR 9: SCHEMATISCHE WEERGAVE BEHEERMODELLEN

De modellen zijn echter geen methode of werkwijzen. Het zijn raamwerken voor de ondersteuning bij de beheersactiviteiten en is een collectie van best-practices op de bijpassende beheergebieden. Door de ondersteuning van een beheermodel kan een organisatie efficiënter beheer toe te passen. (Hoving & van Bon, 2013) (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

3.5.4. ASL

Om het beheeraspect van dit onderzoek te onderzoeken is het goed om een referentiekader te gebruiken en de resultaten van het onderzoek te verifiëren met de best-practice in de praktijk. Hiervoor plaatsen we het onderzoek in een beheeromgeving. Thinkwise bevindt zich, als software ontwikkelaar, op de aanbodzijde van het beheermodel van Looijen (Figuur 8: Beheermodel van Looijen). Als men de bovenstaande beschrijving over de verschillende beheergebieden mee neemt in de positionering van het onderzoek kan de conclusie getrokken worden dat het onderzoek gepositioneerd is in het applicatiebeheer. Thinkwise is verantwoordelijk voor de ontwikkeling en het beheer van de applicatie zelf maar niet voor de inrichting van de infrastructuur van de klant.

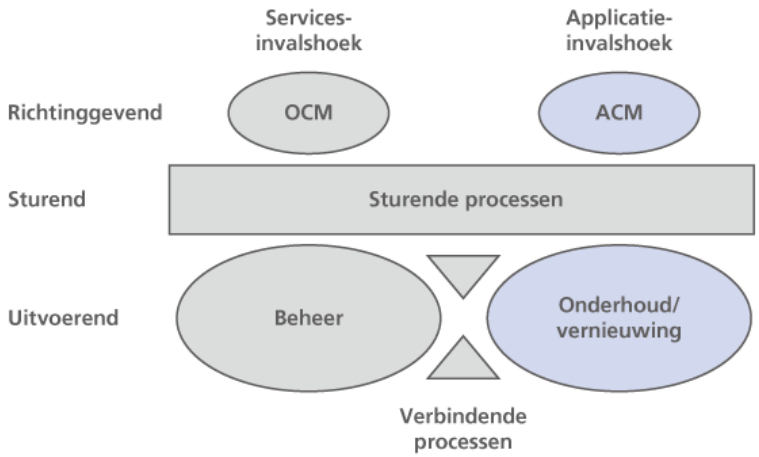
Deze positionering in de beheergebieden bepaald direct ook het referentiekader. De best practice op het gebied van Applicatiebeheer is ASL. ASL heeft dan ook als doel om een public-domain-framework te zijn. Dit bereiken ze door het framework te ontwikkelen met een stichting waar verschillende grote organisaties meewerken en gezamenlijk de best-practices in het beheergebied actualiseren, verbeteren, nieuwe best-practices opvoeren en het framework aan te passen en mee te laten lopen met de ontwikkelingen in de praktijk. ASL heeft dan ook niet als doelstelling om een statisch geheel te laten zijn maar de kennis en ervaringen van organisaties die ermee werken terug te laten vloeien naar ASL. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Het ASL-framework

ASL richt zich op zes clusters van processen zie hiervoor Figuur 10: Het ASL-framework op hoofdlijnen. Ieder procescluster bevat een aantal deelprocessen. De processen in een cluster hebben daarbij een nauwe samenhang en realiseren samen een duidelijk afgebakende doelstelling.

Er zijn zes procesclusters:

1. Beheer
2. Onderhoud en vernieuwing
3. Verbindende processen
4. Sturende processen
5. ACM (Applications Cycle Management)
6. OCM (Organization Cycle Management) (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)



FIGUUR 10: HET ASL-FRAMEWORK OP HOOFDLIJNEN

Beheer

De doelstelling van beheer is ervoor te zorgen dat de huidige applicaties optimaal worden ingezet ter ondersteuning van de bedrijfsprocessen met een minimum aan middelen en verstoringen in de operatie. Hier ligt dan ook de doelstelling van de applicatie. Ze moeten dus 'draaien' en prettig werken. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Onderhoud en vernieuwing

Doelstelling van dit cluster is ervoor te zorgen dat applicaties worden aangepast aan de veranderende wensen en eisen als gevolg voor veranderingen in omgeving en proces. Het resultaat is dan ook dat de applicaties de bedrijfsprocessen optimaal blijven ondersteunen. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Verbindende processen

Programmatuur en data uit bovenstaande clusters wordt via verbindende processen in beheer gebracht. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Sturende processen

De doelstelling van dit cluster is het bewaken dat de bestaande activiteiten conform doelstelling, afspraken en gekozen strategie worden uitgevoerd. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

ACM

Dit cluster heeft als doelstelling om een langetermijnstrategie voor de verschillende applicatieobjecten in het geheel van de informatievoorziening vorm te geven. De geschiktheid van een applicatie of het applicatielandschap wordt vroegtijdig bepaald zodat organisaties niet in de situatie komen dat de informatievoorziening veranderd moet worden. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

OCM

Dit cluster heeft de doelstelling dat het zorg draagt aan de invulling die wordt gegeven aan het beleid en de toekomst van de serviceorganisatie. De toekomstige dienstverlening van de serviceorganisatie wordt hierin bepaald en vervolgens vertaald naar beleid en maatregelen. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Servicegericht
De service-invalshoek richt zich op het leveren van diensten aan personen of organisaties. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Applicatiegericht
De applicatie-invalshoek richt zich op het verkrijgen van expertise in het bedrijfsproces, de klantenwereld ervan, de ontwikkelingen hierin en de feitelijke applicaties. Ook bevat het de veranderingen in het bedrijfsproces en het gestructureerd uitvoeren hiervan. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Uitvoerende processen
De uitvoerende processen zijn het belangrijkste niveau van processen. Zonder deze processen gebeurt er niets en zijn het doel van een applicatiemanagementorganisatie. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Sturende processen
De sturende processen zit in het midden van het model. Ze zijn een knikpunt tussen beleid en operatie. Dit procescluster zorgt ervoor dat de dynamiek van de markt in evenwicht blijft met de strategische verbetering en uitvoerende kwaliteit. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Richtinggevende processen
Een aantal processen hebben een richtinggevend karakter. Het woord richtinggevend wordt daarbij gebruikt om aan te geven dat het een koers is, maar geen heilig doel. Bijstelling zal hierin continu plaatsvinden. De sturende processen vinden in de regel eenmaal per jaar of per twee jaar plaats. Er wordt gestructureerd naar de huidige situatie en de ontwikkelingen in de buitenwereld gekeken en de doelstellingen worden getoetst op haalbaarheid. Op basis van deze informatie worden nieuwe doelstellingen opgesteld. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

3.5.5. Kwaliteitsmanagement in ASL
In ASL heeft kwaliteitsmanagement als doelstelling zorg te dragen voor de (interne en ingekochte) kwaliteit van proces, product, middelen en organisatie door deze te definiëren en te bewaken en ook zorg te dragen dat relevante regelgeving op dit terrein wordt geïmplementeerd en gevolgd. Tevens gaat kwaliteitsmanagement hier om het bepalen van de mogelijke en gewenste verbetermogelijkheden en het zorgdragen dat deze worden gerealiseerd. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

Onderwerpen
Kwaliteitsmanagement richt zich op een viertal onderwerpen:

- Kwaliteit van het geleverde product. Aspecten daarbij zijn kwaliteit van gegevensstructuur en kwaliteit van documentatie.
- Kwaliteit van het (voortbrengings) proces, zoals de kwaliteit van de inrichting van processen, rollen, verantwoordelijkheden en procedures.
- Kwaliteit van de organisatie. Dit heeft als onderwerp de kwaliteit van mensen, expertises, plaats in de organisatie, competities, cultuur, etc.
- Kwaliteit van het kwaliteitssysteem. Onder kwaliteitssysteem verstaan we applicatieontwikkelings- en applicatiemanagementinfrastructuur in brede zin zoals tooling, hulpmiddelen, methoden en technieken. (van der Pols, ASL 2 - Een framework voor applicatiemanagement, 2009)

3.6. Verandermanagementmodellen

Aangezien er naar aanleiding van dit onderzoek een aantal veranderingen plaats zullen vinden, is het ook belangrijk om de medewerkers te betrekken bij de toe te passen veranderingen. Het gaat hier dus om de menselijke kant van de implementatie van de verbeteringen.

3.6.1. Het WWK-model

Het WWK-model is een bewezen model welke bedoeld is om de weerstand van medewerkers in een organisatie op aanstaande veranderingen te voorkomen. Volgens het WWK-model gaat het erom dat de medewerkers begrijpen waarom de innovatie, vernieuwing of verbetering wordt doorgevoerd en wat dit voor hun betekend (weten). Als de medewerkers betrokken worden en vertrouwen hebben in deze aanpassing neemt de kans op succes toe (willen). Door ruimte te geven en vertrouwd te raken met de doorgevoerde wijzigingen neemt de veranderbereidheid in de toekomst verder toe (kunnen).

FIGUUR 11: HET WWK-MODEL

Toepassing

Het WWK-model kan worden ingezet bij de volgende voorbeeldsituaties:

- Een verbetertraject waarbij veel onbegrip en weerstand heerst. Door het WWK-model worden medewerkers betrokken en kan het traject soepeler verlopen.
- Het creëren van draagkracht bij bepaalde beslissingen in een organisatie.
- Het opmerken van fouten in verbetertrajecten doordat de gehele organisatie mee denkt wat dit voor de praktijk betekent. (Nooij, 2015)

4. METHODE VAN ONDERZOEK

Dit onderzoek richt zich op het kwalitatief verbeteren van de software lifecycle van Thinkwise. In dit hoofdstuk wordt een verduidelijking gegeven van de wijze waarin het onderzoek heeft plaatsgevonden. Om dit zo goed mogelijk te verduidelijken is eerst het type onderzoek beschreven, vervolgens wordt er uitgelegd hoe de informatie is verzameld en ten slotte wordt er per deelvraag een beschrijving gegeven hoe de resultaten tot stand zijn gekomen en waarom deze specifieke informatie in het theoretisch kader is opgenomen.

4.1.1. Type onderzoek

Voor dit onderzoek is gekozen om een kwalitatief onderzoek te verrichten. Het betreft een onderzoek naar de huidige software lifecycle en de inzichten en ervaringen van de medewerkers van Thinkwise. Dit is vervolgens vergeleken met gerelateerde literatuur. Aan de hand van die vergelijking zijn er verbeteringen ontworpen. De onderzoeksfuncties zijn dan ook ‘vergelijkend’, ‘beschrijvend’, ‘definiërend’ en ‘ontwerpend’.

4.1.2. Informatievergaring

Aangezien er in dit onderzoek veel vergelijkingen met de literatuur plaats hebben gevonden, is er gekozen om twee methodieken van informatieverzameling te combineren. De 5-stappen methode en de sneeuwbalmethode (Hogeschool Rotterdam, 2014). De 5-stappen methode is een methode om in 5 stappen relevante literatuur te vinden, te beoordelen op relevantie, bruikbaarheid en betrouwbaarheid. De methode splitst zich op in 5 vragen: Wat zoek ik?, Waar zoek ik?, Hoe zoek ik?, Wat heb ik? en Verfijnen. De sneeuwbalmethode is het creëren van een sneeuwbaleffect door via een valide informatiebron verder in te zoomen op de daarbij gebruikte verwijzingen. Zo blijft men in een relevante omgeving. Om de informatieverzameling te beschrijven kan ook de 5-stappen methode uitgelegd worden:

- Wat zoek ik?
  Door alle vragen op te splitsen en begrippen te filteren zijn de volgende zoektermen gedefinieerd: Kwaliteit, Definitie kwaliteit, Kwaliteitsmanagement, Kwaliteitsmanagementmethoden, Software Development Lifecycle, Software Development Lifecycle modellen, Best-practice Software Development Lifecycle, IT-Alignment, Best-practice procesinrichting, Beheermodellen, ASL, verandermodellen, WWK-model
- Waar zoek ik?
  Informatie is gezocht op google, google scholar, de Saxion bibliotheek, de interne bibliotheek van Thinkwise en via de relevante lesstof op Blackboard.
- Hoe zoek ik?
  Door de bovengenoemde specificering van de opgestelde begrippen is er gezocht naar bronnen met relevante informatie. Hiervoor is het register bij fysieke boeken gebruikt of de zoekfunctie bij digitale versies. Door de sneeuwbalmethode te gebruiken is er getracht triangulatie toe te passen en meer relevante informatie te vinden.
- Wat heb ik?
  Vergaarde informatie uit het literatuuronderzoek, die nodig zijn voor het beantwoorden van de deelvragen, zijn samengevat en eventueel getrianguleerd met een derde bron. Deze zijn vervolgens opgenomen in het Theoretisch Kader met een bijbehorende verwijzing naar de literatuur.

- Verfijnen
  De verschillende gevonden informatie wordt vervolgens vergeleken in de resultaten en verfijnt waardoor het toepasbaar is op het onderzoek en de situatie van Thinkwise.

4.1.3. Onderzoeksopzet

In dit hoofdstuk wordt er per deelvraag beschreven welke methodieken er zijn gebruikt om tot beantwoording van de deelvragen te komen. De deelvragen en sub-deelvragen zijn beschreven in hoofdstuk 2.3 - Onderzoeksvragen.

1. Wat wordt er verstaan onder kwaliteit en hoe wordt dit beheerst in een organisatie?

Deze deelvraag is de start van het onderzoek. De onderbouwing die hierbij gebruikt is, is dat het onderzoek zich richt op het kwalitatief verbeteren van de software lifecycle. Om te onderzoeken hoe iets kwalitatief verbeterd kan worden moet men eerst vaststellen wat kwaliteit inhoudt.

Deskresearch

Allereerst is er literatuuronderzoek gedaan naar de definitie van kwaliteit. Om het in context te plaatsen is ook nagegaan hoe dit in een organisatie wordt beheerst. Deze informatie is opgehaald doormiddel van deskresearch. De wijze van informatieverzameling is te vinden in hoofdstuk 4.1.2. De gevonden informatie zoals definities en gebruikte kwaliteitsmanagementmethoden zijn voor een goede vergelijking opgenomen in het theoretisch kader. Aan de hand van de definitievergelijking is er in een definitie van kwaliteit vastgesteld en een vergelijking gedaan van kwaliteitsmanagement.

Semigestructureerde Interviews

De resultaten van het literatuuronderzoek zijn vervolgens gelinkt met Thinkwise. Doormiddel van het houden van interviews is er een beeld geschetst van hoe Thinkwise kwaliteit definieert en beheerst. De keuze op een semigestructureerd interview is gevallen om het interview wel kaders te geven en de interviewer na te laten denken in een vooraf bepaalde richting en zo tot eigen inzichten te laten komen.

Vervolgens is er een vergelijking gemaakt met de literatuur en de praktijk om zo tot een conclusie te komen.

2. Wat maakt een software lifecycle succesvol?

Na het onderzoeken van het begrip ‘kwaliteit’ is de software lifecycle onderzocht. Dit is tevens een onderdeel van het vooronderzoek. Er is onderzoek gedaan naar de definitie van een software lifecycle en naar best-practices op de markt.

Deskresearch

Om tot deze resultaten te komen is er weer deskresearch uitgevoerd. De wijze van informatieverzameling is te vinden in hoofdstuk 4.1.2. De gevonden informatie zijn ook hier voor een goede vergelijking opgenomen in het theoretisch kader. Uit het literatuuronderzoek is een conclusie gesteld dat de waterval-, de iteratieve- en de agile-methode voor dit onderzoek worden gebruikt als referentiekaders. De wijze waarop deze keuze is gebaseerd is beschreven in de inleiding van hoofdstuk 3.3.2. De gekozen best-practices en de definitie van een software lifecycle zijn opgenomen in het Theoretisch kader en vervolgens in de resultaten met elkaar vergeleken om zo tot een antwoord van de deelvraag te komen. Tevens is de literatuurstudie vooraf essentieel geweest voor het opdoen van de kennis over een software lifecycle waardoor er in latere interviews meer sturing gegeven kon worden en er gerichtere vragen in een goede context gesteld konden worden.

3. Wat is de huidige kwaliteit van de gehanteerde Software Lifecycle?

Nadat het vooronderzoek heeft plaatsgevonden en er een beeld is geschetst van het begrip ‘kwaliteit’ en een software lifecycle, is de volgende stap om de huidige situatie in kaart te brengen. Je kan immers pas in beeld brengen wat en hoe er verbeterd moet worden als je een goed beeld hebt van de huidige stand van zaken.

Semigestructureerde Interviews
Om te onderzoeken hoe de lifecycle bij Thinkwise wordt gehanteerd zijn er verschillende interviews gehouden met projectleiders en senior-medewerkers van Thinkwise. De interviews werden gehanteerd door middel van de 5x waarom methode. De keuze voor een semigestructureerd interview bij deze deelvraag is om de medewerker ook richting te laten geven in de antwoorden en actief mee te laten denken om zo tot nieuwe inzichten te komen of het interview zelf sturing te geven als dit vanuit zijn expertise een goede keuze lijkt.

5x-waarom methode
Tijdens de semigestructureerde interviews is er gebruik gemaakt van de 5x waarom methode om zo tot de kern van eventuele problemen te komen. De 5x waarom methode is een Lean Six Sigma tool om oorzaken (root-cause) van een probleem te identificeren. Het identificeren van de oorzaak van een probleem in het proces is een belangrijk doel van de interviews. Met deze methode wordt het mogelijk om tijdens de interviews dieper in te gaan op een bepaalde inefficiëntie of probleem om de kern van de oorzaak te achterhalen.

Procesanalyse
Aan de hand van de verkregen informatie uit de interviews en de bestaande interne documentatie is er een software lifecycle opgesteld. Daarbij zijn ook de onderliggende processen in kaart gebracht. De software lifecycle is op hoog niveau vergeleken met de best-practices die zijn behandeld in hoofdstuk 2. Om de onderliggende processen te onderzoeken is een procesanalyse uitgevoerd. Hiervoor is ASL als referentiekader gebruikt in de vorm van een literatuurstudie. Dit omdat het een best-practice is voor het beheersen van processen in de applicatiemanagement-kant. Thinkwise fungeert daarbij als aanbieder van een informatiesysteem waardoor ASL toepasbaar wordt in dit onderzoek. Om deze reden is een beschrijving van ASL opgenomen in het theoretisch kader.

Voor het beantwoorden van de deelvraag is de kwaliteit van de huidige software lifecycle en haar onderliggende processen getoetst met de behandelde best-practices in deelvraag 2 en de literatuur van ASL.

4. Hoe kan de kwaliteit van de software lifecycle verbeterd worden?

Als de huidige situatie in kaart is gebracht, is het zaak om verbeteringen inzichtelijk te krijgen. Hiervoor zijn de resultaten uit de vergelijking in deelvraag 3 als input gebruikt. De verbeteringen worden hiervoor opgedeeld in drie categorieën: procesmatige verbeteringen, alignmentverbeteringen en de toepasbaarheid van een dashboard.

GAP-analyse
In deelvraag 2 werd er een analyse uitgevoerd op de huidige lifecycle. Hierbij werd de kwaliteit van de onderliggende processen in de software lifecycle getoetst met ASL. In dit hoofdstuk is het zaak om de verbeteringen in kaart te brengen. Hiervoor is een GAP-analyse uitgevoerd waarin het verschil tussen de huidige processen en ASL in kaart werden gebracht. Door het in kaart brengen van deze

‘gaps’ wordt duidelijk waar in het proces er verbeteringen plaats moeten vinden. De verschillen worden vervolgens onderscheiden op procesmatig- of alignmentgebied.

Focusgroep
Om inzicht te krijgen in de kwaliteit van de processen is de opdracht gegeven om een dashboard in te richten. Voor het opstellen van het dashboards zullen er kritieke prestatie-indicatoren (KPI’s) moeten worden opgesteld. Het bepalen van deze KPI’s is gedaan aan de hand van een focusgroep met verschillende projectleiders en senior-medewerkers. Hierbij hebben de medewerkers gezamenlijk de KPI’s opgesteld. Hierbij is tevens de 5x-waarom methodiek gebruikt om te achterhalen waarom een bepaalde KPI moet worden geformuleerd. Na de focusgroep zijn kritieke processen in kaart gebracht en zijn er metingen opgesteld om deze kritieke processen meetbaar te maken.

De resultaten van de focusgroep en de GAP-analyse zijn vervolgens weer met ASL getoetst op validiteit. Er is vervolgens een aanvulling gedaan op de opgestelde metingen aan de hand van deze toetsing in de vorm van een literatuurstudie.

5. **Hoe moeten de benodigde verbeteringen worden geïmplementeerd?**

Nu er verbeteringen in kaart zijn gebracht is het vervolgens nodig om te onderzoeken hoe deze verbeteringen in de organisatie geïmplementeerd kunnen worden. De input hiervoor zijn uiteraard de opgestelde verbeteringen in deelvraag 4. Er is hierbij tevens een scheiding gemaakt tussen procesmatige- en alignmentverbeteringen.

WWK-model
Om het menselijk aspect van procesmatige veranderingen mee te nemen in het onderzoek en de weerstand van medewerkers op deze veranderingen in de organisatie weg te nemen is het WWK-model in context geplaatst. Door literatuurstudie is onderzocht op welke punten het WWK-model zich richt en hoe dit model voor een effectievere veranderstrategie kan zorgen. Door het WWK-model toe te passen op de vastgestelde veranderingen kan Thinkwise inspelen op het risico dat het weerstand krijgt op de door te voeren veranderingen in de projectmethodiek.

Softwareanalyse
Om de toepasbaarheid van de veranderingen in kaart te brengen is er een softwareanalyse gedaan. Dit is uitgevoerd in de vorm van een interview met de manager van de afdeling: Software Modernization. Het interview is uitgevoerd in een gestructureerde vorm omdat het ging om het ophalen van vooraf opgestelde vragen en het boven tafel krijgen van informatie met een duidelijk doel. Het resultaat van dit interview is dan ook een schatting van de benodigde uren en een overzicht van de technische implementatiemogelijkheden. De opgestelde informatie en schatting van de benodigde uren voor realisatie zijn vervolgens geverifieerd bij medewerkers van zowel de afdeling: Product Innovation als Customer Solutions.

De opgehaalde informatie uit de software analyse is ten slotte verwerkt in een stappenplan, welke fungeert als handvat voor de realisatie van de oplossing. Dit uitgeschreven stappenplan is vervolgens verwerkt in een processchema. Het processchema is opgesteld aan de hand van de BPMN 2.0. methode.

5. RESULTATEN

In dit hoofdstuk wordt het onderzoek toegelicht en de resultaten uitgewerkt. De resultaten worden gebundeld en daarmee wordt de hoofdvraag beantwoord. Als eerste worden de resultaten van het vooronderzoek geïntroduceerd om definities vast te leggen, de best-practices in kaart te brengen en te onderzoeken hoe er in de praktijk met deze materie wordt gewerkt. Vervolgens is de huidige Software Lifecycle van Thinkwise door middel van interviews in kaart gebracht en getoetst met de literatuur. Aan de hand van een GAP-analyse worden de verbeterpunten geschetst en onderbouwd. Ten slotte wordt er een beschrijving gedaan van de te ondernemen stappen om de verbeterpunten te implementeren.

5.1. Kwaliteit en Kwaliteitsmanagement

In dit hoofdstuk wordt kwaliteit onder de loep gelegd. Het onderzoek is immers gericht op het verhogen van de kwaliteit van de software lifecycle. Het (gewenste) gevolg van het verhogen van de kwaliteit van de software lifecycle zal dan ook zijn dat de kwaliteit van het eindproduct stijgt. Het benaderen van het begrip 'kwaliteit' is dus van essentieel belang voor het onderzoek. De deelvraag die bij dit hoofdstuk wordt behandeld is:

"Wat wordt er verstaan onder kwaliteit en hoe wordt dit beheerst in een organisatie?"

Kwaliteit is een subjectief begrip. Dit wordt duidelijk als men het begrip tracht te omschrijven. Verschillende antwoorden en omschrijvingen zullen gegeven worden. Om te onderzoeken hoe de kwaliteit van de lifecycle verbeterd kan worden, en hoe dit in praktijk wordt beheerst, zal eerst de definitie vastgesteld moeten worden om te weten waarover er wordt gepraat en om misverstanden over de resultaten te voorkomen. Om tot een zo goed mogelijke vaststelling te komen benaderen we de kwaliteitsdefinitie vanuit de literatuur en vanuit de praktijk en kijken we vervolgens hoe dit in de praktijk wordt beheerst.

5.1.1. Kwaliteit volgens de literatuur

In de literatuur wordt kwaliteit ook op verschillende manieren aangevlogen. Garvin, uitgelegd in hoofdstuk 3.1.1, stelt dat de definitie van kwaliteit afhangt van de mate van absoluutheid en objectiviteit. Aan de hand van deze gegevens kan kwaliteit worden gecategoriseerd in een benadering, waarbij de definitie dus verschilt.

ISO 9001, uitgelegd in hoofdstuk 3.1.3, houdt er een absolute definitie op aan. Hier wordt kwaliteit omschreven als de mate waarop een product of dienst voldoet aan de normen, behoefte, wensen of eisen van de eindgebruiker of klant. In de omschrijving van Garvin is dit de gebruiker-gebaseerde benadering. Er wordt in vergelijking met Garvin echter geen rekening gehouden met verschillende situaties.

De benadering van ISO komt deels overeen met de benadering van Juran. Echter omschrijft Juran nog een benadering. In deze extra benadering stelt Juran dat kwaliteit wordt gezien als vrijheid van afwijkingen of vrijheid van fouten.

5.1.2. Kwaliteit volgens Thinkwise

Tijdens interviews met senior medewerkers en managers is gevraagd wat zij verstaan onder kwaliteit. De beschrijving is echter wel in het perspectief geplaatst van Software Ontwikkeling.

De medewerkers van Thinkwise beleven kwaliteit ook verschillend. Een overlappende benadering is hierbij wel te vinden. Kwaliteit wordt bereikt door de mate waarin gemaakte werk voldoet aan de

wensen van de gebruiker/klant in de tijd waarvan de klant dit verwacht. Bij deze definitie wordt wel vaak de aanvulling gegeven dat het gemaakte werk onder motorkap goed moet werken.

5.1.3. Vergelijking Thinkwise met de literatuur

De benadering van het begrip kwaliteit waarbij kwaliteit wordt behaald als het product voldoet aan de wensen van de afnemer in de tijd dat de afnemer dit verwacht, is in de software development branche veel voorkomend. De branche werkt immers op basis van requirements. Het is dan ook niet onverwachts dat de medewerkers van Thinkwise dezelfde benadering aanhouden. Deze benadering is (deels) terug te vinden in de definitie volgens ISO-9001, de gebruikers-gebaseerde benadering van Garvin en ook Juran schrijft dat kwaliteit wordt bereikt door te voldoen aan de eisen en de wensen van de klant. Het is dan ook te concluderen dat Thinkwise een juiste definitie van kwaliteit aanhoudt.

5.1.4. Definitiebepaling

Aan de hand van de bovenstaande vergelijking wordt er een definitie van het begrip kwaliteit vastgesteld die door Thinkwise wordt geïdentificeerd en is getoetst op de literatuur. Vervolgens plaatsen we deze definitie in de context van het onderzoek. Aangezien de context van het onderzoek zich richt op de kwaliteit van het proces die uiteindelijk moet zorgen voor een hogere kwaliteit van het eindproduct, moeten beide facetten, zowel proces als eindproduct, hierin worden meegenomen.

Aan de hand van de vergelijking in hoofdstuk 5.1.3 is de volgende definitie van kwaliteit vastgesteld:

"Kwaliteit is de mate waarin een product, proces of dienst voldoet aan de eisen en wensen van de afnemer/klant binnen de gewenste tijdskaders."

Als men praat over een proces, is de organisatie in deze context de afnemer van het proces.

De definitie van kwaliteit moet echter nog in de context van het onderzoek worden geplaatst. Het gaat hier om het kwalitatief verbeteren van de software lifecycle. Een kwalitatieve lifecycle kan dan omschreven worden als:

"De kwaliteit van de software lifecycle is de mate waarin de software lifecycle voldoet aan de eisen van het kwaliteitsmanagement van de organisatie binnen de gewenste tijdkaders"

In het kwaliteitsmanagement van de organisatie worden de eisen van de lifecycle bepaald. In hoofdstuk 3.5.5 (Kwaliteitsmanagement in ASL) wordt uitgelegd waar de kwaliteitsmanagement in de best-practice van ASL verantwoordelijk voor is. Hier wordt zowel de kwaliteit van het eindproduct als de kwaliteit van het proces beheerst. De verwachting en de eisen van de afnemer zijn dus onderdeel van de eisen voor de lifecycle die door het kwaliteitsmanagement van de organisatie worden opgesteld. De tijdkaders wordt per project bepaald in overeenstemming met de klant. Bij ASL is dit onderdeel van het proces: Contractmanagement. Een goede samenwerking tussen deze twee processen is dus essentieel als men de bovenstaande definitie van kwaliteit aanhoudt.

5.1.5. Het managen van kwaliteit

De definitie van een kwaliteitsmanagementsysteem is te vinden in 3.2.2 (Kwaliteitsmanagementsysteem volgens ISO 9001). Het stelt dat het hier gaat om een doelmatige manier van werken wat ertoe leidt dat de organisatie goede producten en diensten levert waar de klanten tevreden over zijn. Kwaliteitsmanagement wordt door ISO-9001 als volgt gedefinieerd (beschreven in hoofdstuk 3.2.1: Kwaliteitsmanagement):

"*Alle activiteiten binnen het totale management-orgaan die die kwaliteitseisen, doelen en verantwoordelijkheden bepalen en deze implementeren door kwaliteits-planning, -controle en -verbeteringen toe te passen*" (Stichting Nederlands Normalisatie-Instituut , 1994)

In relevante literatuur wordt er regelmatig aangehaald dat het managen van kwaliteit een continu proces is. Het is het combineren en optimaal gebruiken van verschillende facetten in een organisatie waarbij de aandacht voor de processen centraal staat. De centrale benadering op de processen is een terugkomend gegeven in vrijwel alle kwaliteitsbenaderingen. De continue verbetering van de kwaliteit wordt hierbij altijd in relatie gelegd met de processen, en de processen liggen continu onder de loep. TQM (beschreven in hoofdstuk 3.2.3) is een voorbeeld van een kwaliteitsmanagement-methode die verschillende facetten en aandachtspunten met elkaar combineert om zo tot volledige kwaliteit over de gehele organisatie te komen. Onderdeel daarvan zijn; continue verbetering, processen staan centraal en een geïntegreerd systeem als ISO-9000 (beschreven in hoofdstuk 3.2.4).

Als we kijken naar ASL (beschreven in hoofdstuk 3.5.4), de best-practice op het gebied van applicatiemanagement, wordt er ook aandacht besteed aan kwaliteitsmanagement (beschreven in hoofdstuk 3.5.5). Het is hierbij onderdeel van het middelste procescluster (sturend). Het houdt zich bezig met de kwaliteit van het geleverde product, het proces, de organisatie en het kwaliteitssysteem. Kwaliteitsmanagement heeft dan ook als doelstelling om verbetermogelijkheden van alle processen in de organisatie in de organisatie op te merken, te meten en te verbeteren door evaluaties uit te voeren en verbeteringen te initiëren in andere processclusters van de organisatie. Tevens heeft het als doelstelling zorg te dragen dat de relevante regelgeving op dit terrein wordt geïmplementeerd en gevolgd. De mate waarin de geleverde oplossing voldoet aan de afgesproken dienstverlening (dus ook de geleverde oplossing door de leverancier) valt ook binnen de vraagstukken van kwaliteitsmanagement. (van der Pols, Kwaliteitsmanagement, 2009)

Wat opvalt in de vergelijking van deze twee benaderingen van kwaliteitsmanagement is de mate waarin het de kwaliteit over de gehele organisatie meet. Beide methoden hebben als standpunt dat kwaliteit bereikt kan worden als er op alle facetten van de organisatie wordt geëvalueerd. Deze input is essentieel voor het opstellen van de kwaliteitseisen en de totale beheersing van kwaliteit.

5.1.6. Kwaliteitsmanagement bij Thinkwise

Uit interviews is gebleken dat op dit moment de verantwoordelijkheid van het managen van kwaliteit in een project, zowel procesmatig als softwarematig, in de handen ligt van de projectleiders en senior-medewerkers. Dit is echter de kwaliteit van het te leveren product. In de loop van een project worden er afspraken gemaakt met de afnemer van de software over de op te leveren software en de daarbij behorende deadline. Deze deadline wordt door Thinkwise als leidend aangehouden. Dat wil zeggen dat, tenzij de software geen kritieke fouten bevat of de planning dusdanig achterloopt dat deployment niet mogelijk is, de software op het afgesproken moment wordt opgeleverd. Vaak komt het voor dat de software nog niet voldoende is getest, kleine fouten bevat, niet bestand is tegen het gebruik door veel gebruikers of de risico’s op lange termijn niet voldoende in kaart zijn gebracht. Dit is echter geen verassing als er tegelijkertijd wordt aangegeven dat het overzicht van de algemene kwaliteit van het product in een lopend project slecht te vinden is.

Uit de interviews met Robert van Tongeren (senior Thinkwise Specialist) en Harold Soepenberg (projectleider) blijkt dat dit een gevolg is van twee belangrijke aspecten in het projectmanagement. Ten eerste blijkt dat het evaluatiemoment voor belangrijke zaken zoals het opstellen van de requirements of het maken van functionaliteit discutabel is. De evaluatie hierbij is essentieel. Dat wil zeggen dat het noodzakelijk is dat er een evaluatie plaats vindt van het opgeleverde werk voordat men doorgaat met het volgende proces. Het gaat hier echter om mensenwerk. Het werk wordt door een tweede collega gereviewd en goed bevonden. Dit is echter niet altijd de projectleider (en tevens verantwoordelijk voor de kwaliteitsmanagement in het project) zelf. Dit kan ook elke willekeurige tweede collega gebeuren. Hier ontstaat er verlies in controle en overzicht. Het verzamelen van de

resultaten van deze evaluaties, al dan niet op basis van een terugkoppeling van een medewerker, zit in het hoofd van de projectleider. Tevens hangt de validiteit van dit proces af van de expertise van de reviewer en kan de betrouwbaarheid van het controleproces in twijfel worden getrokken. Ten tweede zijn er veel processen waar op het moment geen terugkoppeling of evaluatie plaats vindt. De projectleider moet uitgaan van de expertise van zijn medewerkers of moet alle stappen handmatig controleren als hij het idee heeft dat hier een fout in is gemaakt. Dit gaat ten koste van de efficiëntie.

Zowel Robert-Jan de Nie (projectleider) als Robert van Tongeren vullen hierbij aan dat er ten slotte een hoge prioriteit wordt gegeven aan het bedienen van de klant op de vooraf opgestelde afspraken. De keuze om af te wijken van deze afspraken als de kwaliteit van het eindproduct nog niet voldoet aan de eisen blijft vervolgens uit. Dit heeft tevens te maken met het feit dat het pas laat in het proces voor verassingen komt te staan en de uit te voeren werkzaamheden om die fouten op te lossen niet meer te realiseren zijn voor de deadline. Prioriteiten in de te realiseren oplossingen worden gemaakt of er worden 'loopholes' bedacht om die fouten niet boven water te laten komen. Dit alles om het eindproduct op tijd op te leveren en te voldoen aan de afgesproken deadline.

Er is een correlatie te vinden tussen deze twee ontwikkelingen in het proces. Als de projectleiders tijdens het ontwikkelproces het overzicht in de kwaliteit verliezen komt het logischerwijs bij de afronding van het project tot verassingen te staan. Kwaliteitsmanagement in ASL stelt dat deze verassingen te voorkomen zijn door tijdens de uitvoering van het ontwikkelproces meer inzicht te verschaffen in de algemene kwaliteit en tijdens elke processtap een evaluatiemoment uit te voeren. Dit creëert een tussentijds overzicht in de algemene kwaliteit van het eindproduct waardoor projectleiders sneller in kunnen spelen op afwijkingen van de eisen en kunnen deze afwijkingen sneller met de afnemer worden gecommuniceerd om de aangepaste verwachtingen helder te krijgen (van der Pols, Kwaliteitsmanagement, 2009)

5.1.7. Deelconclusie

Voor dit hoofdstuk is de volgende deelvraag opgesteld en behandeld:

"Wat wordt er verstaan onder kwaliteit en hoe wordt dit beheerst in een organisatie?"

In de literatuur staat veel beschreven over de definitie van kwaliteit. Het is echter afhankelijk van meerdere factoren, waaronder het onderwerp waar de kwaliteit over gaat. Een rode draad in de uiteenlopende definities is het voldoen aan de eisen en wensen van de gebruiker of klant. De mate van kwaliteit wordt dus bepaald door de partij die gebruik maakt van de product of dienst.

Als deze benadering van kwaliteit in de context van het onderzoek wordt geplaatst kan de volgende definitie van kwaliteit met betrekking tot een software lifecycle worden gedefinieerd:

"De kwaliteit van de software lifecycle is de mate waarin de software lifecycle voldoet aan de eisen van het kwaliteitsmanagement van de organisatie binnen de gewenste tijdkaders"

In het kwaliteitsmanagement van de organisatie worden de eisen van de lifecycle bepaald.
De definitie van kwaliteitsmanagement die hierbij aangehouden wordt is: "*Alle activiteiten binnen het totale management-orgaan die die kwaliteitseisen, doelen en verantwoordelijkheden bepalen en deze implementeren door kwaliteits-planning, -controle en -verbeteringen toe te passen*"

In het kwaliteitsmanagement wordt zowel de kwaliteit van het eindproduct als de kwaliteit van het proces beheerst. De verwachting en de eisen van de afnemer zijn dus onderdeel van de eisen voor de lifecycle die door het kwaliteitsmanagement van de organisatie worden opgesteld. De tijdkaders wordt per project bepaald in overeenstemming met de klant.

Als er wordt gekeken naar kwaliteitsmanagement bij Thinkwise zijn er twee zaken die opvallend verschillen met de best-practice ASL. De mate en wijze waarin er evaluaties plaats vinden in het project en de wijze waarin afwijkingen van de kwaliteit worden teruggekoppeld naar de afnemer. Er is een correlatie te vinden tussen deze twee ontwikkelingen in het proces. Als de projectleiders tijdens het ontwikkelproces het overzicht in de kwaliteit verliezen komt het logischerwijs bij de afronding van het project tot verassingen te staan. Om deze verassingen te voorkomen zal Thinkwise tijdens de uitvoering van het ontwikkelproces meer inzicht moeten verschaffen in de ontwikkelingen en tijdens elke processtap een evaluatiemoment uit moeten voeren. Dit creëert een overzicht voor projectleiders in de algemene kwaliteit van het eindproduct waardoor ze sneller in kunnen spelen op afwijkingen van de eisen en kunnen deze afwijkingen sneller met de afnemer worden gecommuniceerd om de aangepaste verwachtingen helder te krijgen.

5.2. Software Development Lifecycle

In dit hoofdstuk wordt de Software Development Lifecycle (verkort: SDLC) behandeld. Na het onderzoeken van het begrip kwaliteit is het ook nodig om te onderzoeken wat een SDLC inhoudt en hoe dit in de praktijk wordt gehanteerd en beheerst. Om deze materie te onderzoeken is de volgende deelvraag geformuleerd:

"Wat maakt een software lifecycle succesvol?"

Om deze deelvraag zo goed mogelijk te beantwoorden wordt er gekeken naar de definitie van een SDLC en vervolgens naar de best-practices in een vergelijkbare markt.

5.2.1. Definitie Software Development Lifecycle

In hoofdstuk 3.3.1 in het theoretisch kader is de definitie van een software lifecycle gegeven. Deze luidt: *"De Software Lifecycle is een workflow proces welke de kernfasen en activiteiten in een ontwikkelcyclus definiëren".* (Wong Tiky, 2016) Een SDLC is gericht op het ontwikkelen van software met de hoogst mogelijke kwaliteit tegen de laagste kosten in de kortste tijd. (Pinheiro, 2018)

5.2.2. Best practices volgens de literatuur

Voor het onderzoeken van de best-practices van Software Development Lifecycles is er literatuurstudie gedaan naar gebruikte modellen in de IT sector. Balaji & Sundararajan stellen in hun onderzoek naar SDLC-modellen dat er in de praktijk verschillende modellen zijn ontwikkeld: Waterval, Spiraal, V-model, Rapid prototyping, Incrementeel en Synchroniseer en stabiliseer. (Balaji & Sundararajan Murugaiyan, 2012). Wong Tiky stelt echter in zijn onderzoek dat de meest gehanteerde lifecycle-modellen in de praktijk bestaan uit: Waterval, Incrementeel en Agile. (Wong Tiky, 2016) Aangezien de waterval en de incrementele methode beiden worden benaderd in de twee onderzoeken en de Agile methode wordt gehanteerd bij Mendix (een directe concurrent van Thinkwise) (van der Hoek, 2018), worden deze drie modellen opgenomen in het theoretisch kader en als referentiekader gebruikt voor de vergelijking en het beantwoorden van de deelvraag. Om deze methoden zo goed mogelijk te vergelijken worden deze op drie punten onderzocht.

Toepasbaarheid

De invulling van een software lifecycle is volledig afhankelijk van de invulling van de processen in een project en de voorwaarden van de projectgrenzen, scope, communicatie en manier van productoplevering. Een organisatie kan hier haar eigen invulling aan geven. Succesvolle toepasbaarheid valt dus altijd samen met de inrichting van het operationele deel van de organisatie.

Hantering en fasen

Toch zijn er, ongeacht welke invulling de software lifecycle ook heeft, overlappende fasen te herkennen tussen de verschillende modellen en methodieken. Zowel plannen, ontwerpen,

implementeren, testen en uitrollen zijn allemaal fasen die in vrijwel alle lifecycle modellen terug komen. Het verschil zit hier in de manier waarop het model wordt gehanteerd. De gedachte van een software lifecycle is, in de meeste gevallen, dat het een doorlopende cyclus is waarbij pas na de laatste stap de eerste weer wordt geïnitieerd. Bij de waterval-methode (beschreven in hoofdstuk 3.4.1) is dit bijvoorbeeld het geval. De lineaire invulling van de lifecycle zorgt ervoor dat er pas overgang in fasen plaats vind als de vorige fase volledig is afgerond. Dit maakt het een volwassen methode welke goed te managen is en de organisatie voor minder verrassingen komen te staan aangezien er van te voren duidelijk is gedocumenteerd wat er wordt opgeleverd in de cyclus. Tegelijkertijd maakt dit deze methode een stuk minder flexibel voor tussentijdse aanpassingen of veranderingen in de requirements aangezien de cyclus moet worden afgemaakt voordat men aanpassingen kan maken in voorgaande fasen. (Wong Tiky, 2016) (Balaji & Sundararajan Murugaiyan, 2012)

Dit is bij de iteratieve methode (beschreven in hoofdstuk 3.4.2) echter niet het geval. Het cyclus-component wordt geïnitieerd nadat de requirements in hoofdlijnen voor alle iteraties zijn opgesteld. Een iteratie is een deel van de op te leveren software waarbij in de praktijk niet alle projectleden tegelijkertijd aan één iteratie ontwikkelen. Na de requirement-fase volgt een mini-cyclus waarin software wordt ontwikkeld per iteratie met als eerste stap de designfase waarin de, in de requirement-fase opgestelde, requirements verder worden uitgewerkt voor de ontwikkeling van die specifieke iteratie. Na alle iteraties ontwikkeld te hebben, volgt de deployment van de gehele ontwikkelde software. Aangezien deze methode een kort cycluscomponent heeft, kan het snel reageren op veranderingen in de requirements. Tevens kan er parallel ontwikkeld worden aangezien de requirements op grote lijnen over alle iteraties worden opgesteld. Het verduidelijken van de, op hoofdlijnen opgestelde, requirements van een iteratie valt hierdoor in de ontwikkelcyclus van de iteratie, waarbij het mogelijk wordt om meerdere iteraties tegelijkertijd te ontwikkelen. In tegenstelling tot de watervalmethode, waarbij de cyclus in zijn geheel wordt uitgevoerd en in éénzelfde cyclus alle opgestelde requirements worden ontwikkeld voordat men een nieuwe cyclus in gaat met een nieuwe set requirements. Een nadeel van de iteratieve methode is echter dat de methode, in tegenstelling tot de watervalmethode, lastig te managen is. Het parallel werken en de hogere mate van veranderingen in de requirements zorgt dat dit chaotisch kan worden en de methode meer foutgevoelig wordt. (Wong Tiky, 2016) (Pinheiro, 2018)

De Agile-methode (beschreven in hoofdstuk 3.4.3) is in feite een combinatie van de waterval- en de iteratieve methode. Het breekt de ontwikkeling van een stuk software op in blokken, zoals de iteratieve methode dit ook doet, maar is het opstellen van de requirements onderdeel van het cyclus component. Dit wil zeggen dat, in tegenstelling tot de watervalmethode, de requirements voor één bepaald ‘blok’ worden opgesteld alvorens deze worden ontwikkeld. Dit maakt het van te voren minder duidelijk wat er uiteindelijk opgeleverd wordt, maar net als de iteratieve methode flexibeler voor eventuele tussentijdse veranderingen in de requirements. Het verschil met de iteratieve methode zit hier in de beheersing per iteratie. Door de requirement-fase op te nemen in de cyclus wordt dit meer beheersbaar en zorgt dit voor overzicht. Tevens vindt de deployment-fase in de iteratieve methode plaats na oplevering van alle iteraties en is dit bij de agile-methode onderdeel van de ontwikkelcyclus. De deployment vindt dus bij de agile methode per ‘blok’ plaats waarbij de klant snel een werkend deel van haar software opgeleverd krijgt. Tegelijkertijd zorgt dit voor een groter risico als dit blok niet voldoet aan de eisen. Dit maakt de noodzaak voor goede sturing in deze methode hoog en minder toegankelijk voor de meer complexere projecten. (Balaji & Sundararajan Murugaiyan, 2012)

Een succesvolle hantering van een software lifecycle methode is dus afhankelijk van de inrichting en de complexiteit van het project, de ontwikkelmethode en mate van betrokkenheid van de klant. De

methode moet aansluiten bij de keuzes van het projectmanagement om tot een succesvolle beheersing van een software lifecycle te komen. Requirements spelen hierbij de grootste rol. De methodieken zijn in de volgende situaties het meest geschikt:

- **Watervalmethode**
  Bij grote projecten waar de requirements vooraf duidelijk zijn gedefinieerd. Dit is omdat de omvang van een dergelijk project de beheersing lastig maakt. Door gefaseerd en lineair de lifecycle te behandelen wordt het beter beheersbaar.
- **Iteratieve methode**
  Bij de (relatief) grotere projecten waar de requirements over tijd veranderen en waar de afnemer veel betrokken is. Dit omdat de software per iteratie groeit. De afnemer blijft betrokken bij de groei van de software en geeft hier constante feedback op. Het gegeven dat de requirement-fase en de deployment-fase over de gehele software wordt uitgevoerd, maakt dit de betere keuze voor projecten van een relatief grotere omvang omdat dit meer risico's met zich mee brengt.
- **Agile-methode**
  Bij kleinere projecten waar requirements in snel tempo veranderen en de afnemer nauw betrokken is bij de ontwikkeling. Door het opleveren van 'blokken' software moet de klant nauw betrokken zijn. De Agile-methode is een snelle en wendbare methode waarbij er snel ontwikkeld en getest kan worden. Dit werkt vaak het beste bij projecten van kleinere omvang. (Balaji & Sundararajan Murugaiyan, 2012) (Wong Tiky, 2016)

Output en documentatie
Als we kijken naar de output van de fasen in de verschillende best-practice methodes, is er geen verschil in outputdocumenten. Dezelfde documenten en specificaties worden bij dezelfde fasen van de methodes opgesteld. Het verschil zit hier echter in de inhoud en de functie van het document. Dit heeft te maken met de werkwijze van de methode. Bij de watervalmethode wordt de volgende fase pas geïnitieerd als de vorige fase is afgerond. De bijbehorende documentatie is hierbij van essentieel belang omdat dit als noodzakelijke input fungeert voor de volgende fasen. Dat wil tegelijkertijd ook zeggen dat de inhoud van die documentatie veel meer omvat. Het omvat alle benodigde informatie van die fase over de gehele te ontwikkelen scope.

Dit is bij de iteratieve methode niet het geval. Hier worden alleen de requirements over de gehele applicatie in grote lijnen bepaald, en wordt de documentatie vervolgens per iteratie opgeleverd. De functie van de benodigde documentatie wordt hierdoor minder essentieel omdat de methode ook gericht is op het inspelen op tussentijdse veranderingen. De inhoud van de documentatie is dus ook per definitie minder omvattend, aangezien dit over een iteratie gaat en niet over de gehele scope of applicatie.

De Agile methode komt qua documentatieoplevering overeen met de beide methodes. Hier worden, zoals beschreven in bovenstaande analyse, alle fasen in een cycluscomponent afgewerkt. Dit gebeurt echter per module of iteratie. Dit maakt dat de documentatie ook per module wordt opgesteld, wat, net als bij de iteratieve methode resulteert in zeer gelimiteerde documentatie per iteratie.

5.2.3. Deelconclusie
Voor dit hoofdstuk is de volgende deelvraag geformuleerd:

"Wat maakt een software lifecycle succesvol?"

De Software Lifecycle, ook wel Software Development Life Cycle (verkort: SDLC) is een workflow proces welke de kernfasen en activiteiten in een ontwikkelcyclus definiëren (Wong Tiky, 2016). Het is gericht op het ontwikkelen van software met de hoogst mogelijke kwaliteit en de laatste kosten in de kortste tijd. Het omvat een gedetailleerd plan voor het ontwikkelen, aanpassen, onderhouden en vervangen van een informatiesysteem. (Stackify, 2019)

Een Software Development Lifecycle is altijd afhankelijk van de ontwikkelmethode en werkwijze van een organisatie. De toepassing ervan moet dan ook passen bij de verschillende aspecten die het ontwikkelproces definiëren. Dit omvat zowel de mate waarin de requirements vooraf worden verduidelijkt, de communicatie met en de betrokkenheid van de klant, de omvang van het project en de ontwikkelmethodiek. Een succesvolle software lifecycle is dus op een dusdanige wijze ingericht dat het inspeelt op de afspraken binnen het project en dit tevens aansluit bij de methode waarop er software wordt ontwikkeld.

5.3. De software lifecycle bij Thinkwise

Dit onderzoek is gericht op het kwalitatief verbeteren van de software lifecycle van Thinkwise. Hiervoor is het uiteraard nodig om ook de huidige kwaliteit van de Software Lifecycle in kaart te brengen. Om dit in het volgende hoofdstuk te onderzoeken is de volgende deelvraag geformuleerd:

"Wat is de huidige kwaliteit van de gehanteerde software lifecycle?"

In de afgelopen weken zijn er verschillende interviews gehouden met projectleiders en senior ontwikkelaars bij de afdeling Customer Solutions. Het doel van deze interviews was om inzicht te krijgen in de huidige software lifecycle van Thinkwise en te achterhalen wat er mogelijkerwijs verbeterd kan worden. Aan de hand van de verkregen informatie is er een software lifecycle in kaart gebracht. Deze is te vinden in bijlage 4. In dit hoofdstuk wordt de lifecycle onder de loep genomen. Per fase wordt er een omschrijving gegeven van de activiteiten in die fase, het onderliggende proces en de beheersing.

5.3.1. Hantering

Waar de analyse-, uitrollen en onderhoudsfase over de gehele applicatie worden uitgevoerd, vindt er een doorlopende cyclus plaats van de fasen: requirements tot testen. Dat wil zeggen dat elke module deze 6 stappen doorloopt en de cyclus ook wordt geïnitieerd als de testen niet succesvol zijn of er requirements aangepast worden. Het cycluscomponent wordt pas gestart als de analysefase is afgerond en de uitrolfase als alle modules, en dus de gehele applicatie, zijn ontwikkeld.

<u>Vergelijking met best-practice</u>

Als de software lifecycle van Thinkwise wordt vergeleken met de best-practices als de Watervalmethode (beschreven in hoofdstuk 3.4.1), de Iteratieve methode (beschreven in hoofdstuk 3.4.2) of de Agile methode (beschreven in hoofdstuk 3.4.3) dan zijn er op hoog niveau veel overeenkomsten te zien. De lifecycle van Thinkwise is een mix van de agile- en de iteratieve methode. De opzet en de volgorde van de te doorlopen stappen voldoen aan een kwalitatieve lifecycle. Als we kijken naar de verschillende fasen en de volgorde in de software lifecycle zijn er ook sterke overeenkomsten te zien. Thinkwise ontwikkeld vaak in modules waardoor de cyclus in het model ook per module wordt afgewerkt. Dit is zowel bij de iteratieve- en de agile-methode het geval. De analyse vooraf en de Deployment- en onderhoudsfase achteraf is ook terug te vinden in de best-practices. Zodra dit overeenkomt met de werkwijze in de praktijk en de wijze van ontwikkelen, dan kan er gesproken worden van een goede invulling van de software lifecycle. (Pinheiro, 2018) (Wong Tiky, 2016)

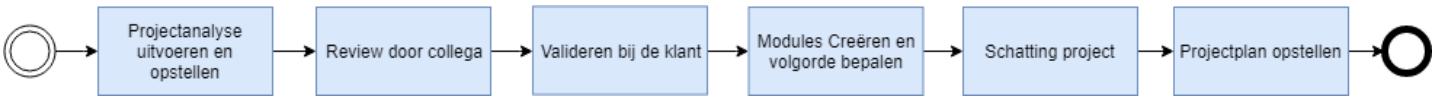
Pas als we inzoomen op de onderliggende processen wordt het referentiekader troebel. Om de kwaliteit van de beheersing hier beter te onderzoeken wordt er een ander referentiekader voor de beheersing van de onderliggende processen gebruikt. Hierbij zijn beheermodellen een oplossing. In hoofdstuk 3.5.4 is uitgelegd wat ASL inhoudt en waarom dit toepasbaar is op het onderzoek. De beheersbaarheid per fase, en dus op de onderliggende processen, wordt dan ook getoetst op ASL.

5.3.2. Analyse

Omschrijving

Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat in de analyse-fase van het project de projectgrenzen worden bepaald en er een schatting wordt gemaakt van de scope, de omvang van het project en de daarbij behorende kosten. In de analyse worden de Business Requirements, de verschillende interfaces vastgelegd en wordt het IT landschap van de klant in kaart gebracht. Uiteindelijk worden de begrippen vastgelegd voor betere beheersing van de klantomgeving. Tevens worden de processen in kaart gebracht wat als uitgangspunt dient voor de software. Aan de hand van de procesregistratie worden de modules bepaald en uiteindelijk een schatting gegeven van de kosten. Met al deze informatie wordt het projectplan opgesteld.

Proces



FIGUUR 12: PROCES ANALYSEFASE

Beheersing

Om de beheersing van het onderliggende proces in deze fase te onderzoeken worden de referentiekaders; Impactanalyse in het procescluster: Onderhoud en Vernieuwing (van der Pols, Impactanalyse, 2009) en Design in het procescluster Onderhoud en Vernieuwing (van der Pols, Ontwerp, 2009) in het beheermodel van ASL gebruikt.

Een aantal processtappen zijn terug te vinden in ASL. Dit is dan ook te vergelijken met de inhoud van de processtappen. De procesvergelijking van deze fase met ASL is te vinden in bijlage 5.

Niet alle processen zijn te onderbouwen met het procescluster: Impactanalyse van ASL. Dit heeft te maken met het eigenaarschap van het proces. Thinkwise heeft, als software leverancier, te maken met afnemers van de software. De afnemer is, in deze zin, voor een deel verantwoordelijk voor de impactanalyse van een wijziging van haar eigen bestaande applicatieprogrammatuur. De splitsing in verantwoordelijkheid die hier gemaakt wordt zijn de activiteiten die betrekking hebben op de doorvoering van de wijzigingen van de applicatie bij de afnemer. De terugkoppeling naar eigen beheerprocessen en infrastructuurmanagement zijn daarbij dus de verantwoording van de afnemer. (van der Pols, Ontwerp, 2009)

De processtap: Review door collega wordt niet genoemd in dit procesdeel van ASL. Toch wordt het regelmatig aangehaald in het proces Kwaliteitsbewaking in ASL (beschreven in hoofdstuk 3.5.5). Het toevoegen van deze processtap brengt geen conflict ten opzichte van het proces volgens ASL en zorgt voor een extra controle op het proces en betere beheersbaarheid van de kwaliteit. Verder lijken alle processtappen in deze fase van de lifecycle voldoende overeen te komen met verschillende delen van ASL om te spreken van een goede beheersing van deze fase.

Het resultaat van de impactanalyse in ASL is te vergelijken met de projectanalyse bij Thinkwise. De inhoud van dit document is ook besproken in de focusgroep. Veel componenten komen overeen,

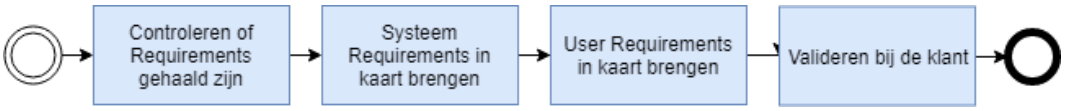
behalve de risico's op korte en lange termijn en de consequenties op de gebruikersomgeving, applicatiemanagement en exploitatieomgeving. (van der Pols, Impactanalyse, 2009) De afweging is hier echter of Thinkwise verantwoordelijk is voor deze analyse of dat dit in handen ligt van de klant.

5.3.3. Requirements

Omschrijving

Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat als men uit een vorige cyclus komt, er dan in deze fase eerst wordt gecheckt of de requirements van de vorige cyclus gehaald zijn. Is dit niet het geval, dan worden er, aan de hand van de in de Analysefase-opgestelde Business requirements, user-requirements opgesteld. Deze requirements dienen als basis voor de uit te voeren werkzaamheden. Deze worden per module opgesteld. Dit maakt het behapbaar en gemakkelijk te managen. Tevens worden er systeem-requirements gerealiseerd. Nadat de klant dit heeft goedgekeurd kan men verder gaan met de volgende fase. (Thinkwise Software B.V., 2019)

Proces



FIGUUR 13: PROCES REQUIREMENTS-FASE

Beheersing

Om de beheersing van deze fase te toetsen worden de processen Ontwerp (van der Pols, Ontwerp, 2009) en Realisatie (van der Pols, Realisatie, 2009) in het procescluster Onderhoud en Vernieuwing van het beheermodel ASL als referentiekader gebruikt. Dit omdat het opstellen van requirements in ASL geen aparte processtap is, maar activiteiten rondom het opstellen van deze requirements te vinden zijn in verschillende processtappen. Om te onderzoeken of deze processtappen in de best-practice ook voorkomen moet de informatie uit beide processtappen komen.

De procesvergelijking van deze fase met ASL is te vinden in bijlage 6.

ASL kent dus geen aparte processtap voor wat betreft het opstellen van requirements. Wél voor ontwerp. Hierin kunnen dus zowel het opstellen van de requirements als het opstellen van de designspecificaties vallen. Thinkwise maakt daarbij onderscheid in Systeem- en User-requirements. Om deze reden zijn deze twee processen als referentiekader gebruikt. Wat opvalt is dat het opstellen van de technische documentatie in het beheermodel van ASL pas na de functionele specificaties komt. Het opstellen van de functionele specificaties binnen ASL gebeurt in het opwerpproces en de technische specificaties in het realisatieproces. Wil Thinkwise hier de best-practice volgen, dan zal het deze twee stappen om moeten draaien.

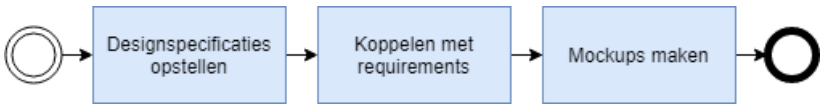
Verder zijn alle beschreven activiteiten in beide processtappen binnen ASL met betrekking tot het opstellen, verwerken en documenteren van specificaties onderdeel van het Thinkwise proces. Hierbij zijn de design-specificaties weggelaten want die vallen binnen de software lifecycle van Thinkwise in een andere fase van de lifecycle. Ondanks dat de volgorde van de processen in de best-practice verschillen ten opzichte van de volgorde van de processen in de praktijk bij Thinkwise, elke activiteit, op de activiteiten: 'bepalen oplossingsrichting' en 'uitwerken oplossingsrichting' na, wordt opgevangen in deze fase van de software lifecycle. De hiervoor genoemde activiteiten zijn onderdeel van de volgende fase van de software lifecycle van Thinkwise. Er kan dus ook hier gesproken worden van een goede beheersing van de processen in deze fase.

5.3.4. Design

Omschrijving

Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat aan de hand van de requirements er designspecificaties worden gemaakt. Deze beschrijven wat het systeem moet doen en hoe dit moet worden gedaan. Dit wordt als handvat gebuikt voor de ontwikkelaars om een duidelijker beeld te schetsen van de te maken software. De designspecificaties worden ook gekoppeld aan de user-requirements. Eventueel worden door projectleiders Mock ups gemaakt om de designspecificaties te verduidelijken.

Proces



FIGUUR 14: PROCES DESIGNFASE

Beheersing

Aangezien designspecificaties binnen de werkwijze van Thinkwise een aparte stap is in het ontwikkelproces en het binnen het framework van ASL onderdeel is van ontwerp, kunnen de processen die onder de designfase van de lifecycle vallen goed worden getoetst. Het enige referentiekader wat hierbij nodig is, is de processtap: Ontwerp (van der Pols, Ontwerp, 2009).

De procesvergelijking van deze fase met ASL is te vinden in bijlage 7.

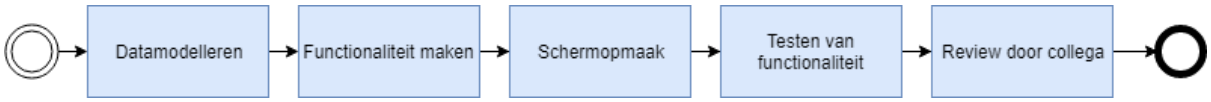
Designspecificaties zijn duidelijk onderdeel van de processtap ontwerp in het beheermodel van ASL. Er is dan ook weinig ruimte voor een verkeerde interpretatie van de activiteiten die horen bij het opstellen van designspecificaties of een Mock up. Wat direct opvalt is dat het koppelen van de designspecificaties met de requirements niet wordt aangehaald in ASL. Echter, de doelstelling van deze processtap in ASL is “De (gebruikers)specificaties van het informatiesysteem of de wijzigingen op een dusdanige wijze op te zetten en vast te leggen, dat deze daarna op een eenduidige wijze kunnen worden gerealiseerd en getest.” (van der Pols, Ontwerp, 2009) Hieruit kan geconcludeerd worden dat het ontwerp is afgeleid uit de specificaties van de afnemer. Robert van Tongeren stelt in een interview dat deze letterlijk met elkaar gekoppeld kunnen worden in de SF. Dit maakt het een processtap in het ontwikkelproces. Concluderend uit bovenstaande analyse kan er ook bij de designfase van de software lifecycle gesproken worden van een goede beheersing.

5.3.5. Ontwikkelen

Omschrijving

Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat nadat de requirements en designspecificaties zijn opgesteld er ontwikkeld kan worden. Bij Thinkwise valt hier zowel het datamodelleren, het schrijven van functionaliteiten, het opmaken en instellen van de schermen en de GUI en het maken van de lay-out onder dit proces. De gemaakte code wordt eerst door de maker getest in de GUI en wordt vervolgens gereviewd door een collega. Dit is geen geautomatiseerde test, maar is enkel gebaseerd op de designspecificaties of requirement en wordt ‘handmatig’ getest. Pas als het ontwikkelen in deze module is afgerond gaat men door naar de volgende fase. (Thinkwise B.V., 2017)

Proces



FIGUUR 15: PROCES ONTWIKKELFASE

Beheersing

Net als de fase Design is ook de fase Ontwikkelen één op één terug te vinden in het beheermodel van ASL (Realisatie) (van der Pols, Realisatie, 2009). Dit is dan ook de enige processtap die als referentiekader wordt gebruikt.

De procesvergelijking van deze fase met ASL is te vinden in bijlage 8.

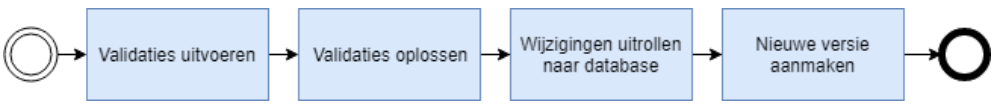
Allereerst is het goed te benoemen dat binnen het proces realisatie geen duidelijk voorgeschreven ontwikkelproces wordt gedefinieerd. Dit wil zeggen dat er als activiteit: 'aanpassen van de programmatuur' wordt verduidelijkt als activiteit bij het realiseren van de oplossing. Tevens stelt ASL dat de activiteiten vaak mede worden bepaald door de gebruikte methode en ontwikkelprogrammatuur. Dit maakt dat het dus lastig bepalen is, al helemaal bij een ontwikkelproces zoals Thinkwise die hanteert, of de processtappen bij het ontwikkelen volgens de best-practice worden uitgevoerd. Wél is er een duidelijke volgorde te zien in realisatie, testen en controleren. Testen is daarbij nog onderdeel van het ontwikkelproces. Dit is ook terug te vinden in het ASL framework. De processtap; Review door collega is, net als bij de analysefase, niet onderdeel van het ASL proces. Dit is echter wel terug te vinden in de processtap: kwaliteitsmanagement.

5.3.6. Dataconversie

Omschrijving

Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat het ontwikkelen altijd plaats vindt binnen een tijdelijke omgeving. Dat wil zeggen dat bepaalde wijzigingen vaak geen invloed hebben op de bestaande data van de klant. Als de ontwikkelingen klaar zijn kunnen de wijzigingen in het datamodel en het GUI-model naar de applicatie geschreven worden. In deze fase van de cyclus komen veel controles kijken omdat het in het ergste geval kan resulteren in verlies van data. Als er wijzigingen nodig zijn dan zal dit opgelost moeten worden voordat de wijzigingen naar de database worden geschreven. Als dit alles gedaan is wordt er een nieuwe projectversie aangemaakt waarin de nieuwe wijzigingen plaats gaan vinden. Dit maakt wijzigingenbeheer mogelijk.

Proces



FIGUUR 16: PROCES DATACONVERSIE-FASE

Beheersing

De volgende processtap is Thinkwise specifiek. Dat wil zeggen dat in een normale lifecycle, het proces: Dataconversie geen onderdeel zou zijn, Zeker niet in het midden van de cyclus. Toch is dit het geval in deze lifecycle. Voordat men de wijzigingen doorzet naar de database wordt er eerst door de software een aantal validaties uitgevoerd. Dit is sterk vergelijkbaar met een tussentijdse impactanalyse op softwarematig gebied. De referentiekaders die hiervoor gebruikt worden zijn dan ook de processen; impactanalyse (van der Pols, Impactanalyse, 2009), wijzigingsbeheer (van der Pols,

Wijzigingenbeheer, 2009) en programmabeheer en distributie (van der Pols, Programmabeheer en distributie, 2009).

De procesvergelijking van deze fase met ASL is te vinden in bijlage 9.

Dat het ontwikkelproces van Thinkwise in de regel veel verschilt met een meer gestandaardiseerde vorm van ontwikkelen is ondertussen duidelijk. De fase Dataconversie is daar een uitstekend voorbeeld van. Door het ontwikkelen in een testomgeving en het uitvoeren van (relatief) kleine stukken programmatuur, heeft Thinkwise een beheersbaar stuk wijziging- en programmabeheer ingebouwd in de software.

Wat opvalt is dat in het beheermodel van ASL het proces; Programmabeheer en distributie in een andere procescluster valt. Dit wil zeggen dat het in het beheermodel van ASL een apart proces is, waar het dient als een productsluis met andere processclusters zoals infrastructuurmanagement. Dit heeft ook te maken met het feit dat ASL zich richt op de distributie van afgeronde programmatuur, in vergelijking met de (vaak onvolledige) brokken programmatuur in de software van Thinkwise.

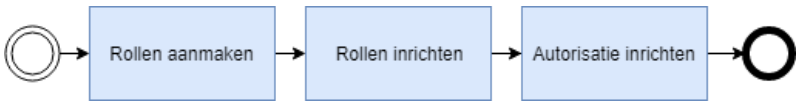
Tevens heeft het procescluster Programmabeheer en distributie ook de doelstelling om de applicatieobjecten (of de informatie daarover) op het juiste moment ter beschikking te stellen aan de juiste processen. (van der Pols, Programmabeheer en distributie, 2009) Dit deel gaat over de alignment van het ontwikkelproces met de rest van het gehele beheermodel en niet enkel in hetzelfde ontwikkelproces. Om deze beter beheersbaar te krijgen zal Thinkwise de verschillende processclusters op de eigen situatie in kaart moeten brengen en de activiteiten daar onder moeten schalen. De vraag is echter of de processclusters dan nog volledig zijn, aangezien het activiteiten uitvoert op, volgens ASL, in de verkeerde processclusters op een verkeerd moment.

5.3.7. Autorisatie

Omschrijving

Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat nadat de dataconversie is afgerond en de wijzigingen zijn doorgevoerd de applicatie wordt ingericht op de gebruikers. Hier worden de rechten voor de gebruikers bepaald. Aan de hand van de gebruikers van de klant, de daarbij behorende rollen en rechten wordt de omgeving en de autorisatie voor de gebruikers ingericht. (Thinkwise B.V., 2017)

Proces



FIGUUR 17: PROCES AUTORISATIEFASE

Beheersing

Net als Dataconversie is ook autorisatie een Thinkwise-specifieke processtap. In veel ontwikkelmethodes is het een apart proces die zich dan ook volgens ASL in een ander procescluster bevind: Namelijk in het proces continuïteitsbeheer (van der Pols, Continuïteitsbeheer, 2009). Het inrichten van de autorisatie is een oorzaak van de discontinuïteit van de applicatie, doordat het de interne beveiliging van de applicatie raakt, en er een risico speelt voor oneigenlijk gebruik van binnenuit. Echter zijn er geen duidelijke activiteiten te vinden in ASL voor het bepalen en inrichten van de autorisatie, maar is dit onderdeel van de analyse en activiteiten behorend bij het proces continuïteit in de procescluster; Beheerprocessen. Hiermee is te concluderen dat de verantwoordelijkheid voor dit proces bij de afnemer ligt. Het aanpassen in de software is dan echter

wel voor rekening van Thinkwise. Die zal dit, volgens ASL, moeten baseren op de input van de afnemer. Dit gebeurt ook, dus kan er gesproken worden van een goede beheersing van de fase.

5.3.8. Testen

Omschrijving

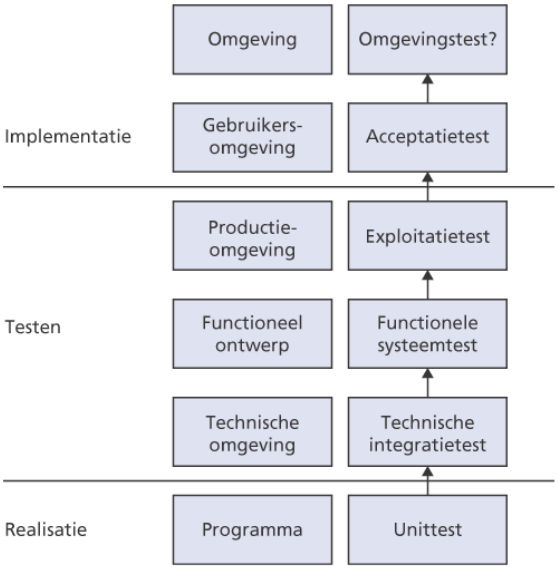
Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat nadat de module ontwikkeld en ingericht is, de applicatie wordt getest. Na het inrichten van de unittest wordt deze softwarematig uitgevoerd. De resultaten hiervan kunnen dus snel en overzichtelijk door de software worden gerealiseerd. Na de unittest wordt er ook een gebruikerstest uitgevoerd. Dit gebeurt aan de hand van de opgestelde bedrijfsprocessen waar de applicatie op is gebaseerd.

Proces

FIGUUR 18: PROCES TESTFASE

Beheersing

Het proces testen is terug te vinden in het procescluster Onderhoud en Vernieuwing in het beheermodel van ASL (van der Pols, Testen, 2009). Dit is in de context van Thinkwise ook waar de Software lifecycle zich bevind. Om die reden wordt het proces Testen dan ook als referentiekader gebruikt bij de vergelijking. Toch is hier een kanttekening te maken: Het proces testen in ASL wordt opgedeeld in verschillende onderdelen. In Figuur 19 is een schematisch overzicht gemaakt van de verschillende testen op de verschillende componenten in het ontwikkelproces. Omdat de implementatie ook pas in de volgende fase wordt uitgevoerd en het overzetten naar de productieomgeving binnen het proces bij Thinkwise pas wordt uitgevoerd na afronding van alle modules, worden deze ook niet meegenomen in het referentiekader voor dit proces:



Figuur 32 Testen in de omgeving

FIGUUR 19: TESTEN IN ASL

De procesvergelijking van deze fase met ASL is te vinden in bijlage 10.

Wat opvalt is dat het Thinkwise-proces: gebruikerstest (het handmatig testen van de requirements) in een andere procesfase valt. Het is onderdeel van het proces Testen, in vergelijking tot het onderdeel Realiseren waar het zich bij Thinkwise in bevindt. Dit heeft te maken met de iteratieve manier van werken binnen Thinkwise en het doorlopen van de fasen per module of gerealiseerd stuk programmatuur welke zich in hoofdstuk 5.2.2 al bewezen effectief heeft bevonden. In het geval van Thinkwise zal een functionele systeemtest (gericht op kleine brokken programmatuur) dus onderdeel worden van het proces Realisatie in het model van ASL.

5.3.9. Uitrollen

Omschrijving

Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat als alle modules af zijn, er een volledige applicatie staat. Deze wordt, als alle requirements zijn gehaald en er aan de wens van de klant voldaan is, in de productieomgeving geplaatst. Om dit soepel te laten

verlopen wordt er een deploymentplan opgesteld nadat er een plan is opgesteld welke stappen er genomen worden om productie mogelijk te maken. In dit proces vind ook de acceptatietest plaats met de daarbij behorende acceptatiedocumentatie. Hier volgen nog enige wijzigingen tot het moment dat de issues nihil worden. Om de overgang naar de afdeling Service & Care goed te laten verlopen wordt er een upgrade-logboek opgesteld met daarin alle informatie en gegevens over de zojuist gerealiseerde applicatie.

Proces

FIGUUR 20: PROCES UITROLFASE

Beheersing

De Deploymentfase (uitrolfase) van de software lifecycle van Thinkwise is niet zozeer Thinkwise specifiek. Het is een deployment-proces die vergelijkbaar is met andere softwareontwikkelingsbedrijven. Er zijn dan ook veel vergelijkingen te vinden met de processtap Implementeren in het beheermodel van ASL (van der Pols, Implementeren, 2009).

De procesvergelijking van deze fase met ASL is te vinden in bijlage 11.

Deze fase van de software lifecycle is de eerste fase waarbij de activiteiten (bijna) één op één overeenkomen met de activiteiten in het vergelijkbare proces in het beheermodel van ASL. Het doel is daarbij ook gelijk, namelijk; Het invullen van de noodzakelijke randvoorwaarden om te komen tot een foutloos gebruik van de nieuwe (versie van de) applicatie.

Waar de processtappen; ‘laatste controle’ en ‘updrade-draaiboek opstellen’ redelijk Thinkwise-specifiek zijn, kunnen ze goed onderverdeeld worden in de activiteiten behorend bij het implementatieproces van ASL. Dit geldt echter ook voor de andere processen in deze lifecycle-fase.

Wat echter wel verschilt van de best-practice ASL is het feit dat Thinkwise zelf de acceptatietest ontwikkelt en uitvoert. Deze conclusie is uit de interviews getrokken. Dit betekent echter dat de acceptatietest zo ontwikkeld wordt, dat Thinkwise de resultaten van deze test vooraf kan incalculeren en voorbereiden. Soepenberg stelt hierbij zelfs dat de acceptatietest in praktijk dus altijd zal slagen. Thinkwise zal, volgens ASL, hierin een ondersteunende factor moeten bieden en de verantwoordelijkheid van het proces in handen brengen van de afnemer waardoor de validiteit van de acceptatietest betrouwbaar wordt. Iets wat dus op het moment niet het geval is.

5.3.10. Onderhoud

Omschrijving

Uit de interviews en de focusgroep met projectleiders en senior-medewerkers is geconcludeerd dat als het uiteindelijk enkel om het oplossen van issues gaat, het project dan wordt overgedragen naar de afdeling Service & Care. Deze ontfermd zich over de verdere onderhoud van de applicatie. Zij hanteren een eigen lifecycle. Omdat deze kan verschillen, er andere rollen mee spelen en tegelijkertijd ook andere processen om de hoek komen kijken, moet er verder gekeken worden naar de interne software lifecycle. De afdeling Service & Care is een afdeling die zich, in termen van ASL, in meerdere procesclusters moet bevinden waardoor een simpele analyse van deze fase van de lifecycle buiten de scope van het onderzoek is gelaten.

5.3.11. Output en documentatie

Veel SDLC best-practices die in de vergelijkbare praktijk worden gebruikt zijn gericht op een gebruikelijk IT bedrijf. Thinkwise wijkt iets af van de gebruikelijke vorm van ontwikkelen omdat het niet 'ontwikkelt' maar 'modelleert'. Toch blijft Thinkwise een Softwareontwikkelingsbedrijf waarbij bepaalde documenten niet gemist kunnen worden. Uit de interviews met verschillende senior-medewerkers en projectleiders is geconcludeerd dat de software lifecycle bij Thinkwise de volgende outputcomponenten bevat:

TABEL 1: OUTPUTVERGELIJKING HUIDIGE SOFTWARE LIFECYCLE

Lifecycle Fase	Output Thinkwise	Output SDLC best-practices	Output ASL
Analyse	Analyse (Business Case) Projectplan	Business Case documentatie	Impactanalyse
Requirements	Systeem-requirements User-requirements	Systeem-requirements User-requirements	Technische specificaties Testplan
Design	Design-specificaties	Ontwerpspecificaties	Ontwerpdocumentatie
Ontwikkelen	-	Functionele specificaties	Functionele specificaties
Dataconversie	-	-	-
Autorisatie	-	-	-
Testen	Testplan	Testplannen	Testresultaten Testverslagen
Uitrollen	Deploymentplan Acceptatietest Upgradedraaiboek	Deploymentplan Calamiteitenplan	-
Onderhoud	-	-	-

Wat opvalt is dat er fasen zijn waar geen output wordt gegenereerd. Dit betekend echter niet dat er geen output is. Door de snelle ontwikkeling van het product en het opnemen van verschillende outputcomponenten in de software zelf, komen er weinig documenten aan te pas. Tevens kan de output ook programmatuur of een nieuwe versie van de software zijn. Dit wordt in het bovenstaande tabel niet als outputdocument beschouwd. Er mist enkel één outputcomponent volgens de SDLC best-practice: het calamiteitenplan. Dit is, in mindere mate, onderdeel van het upgradedraaiboek.

Voor de vergelijking met ASL is er qua outputcomponenten gelet op vergelijkbare documentatie op een vergelijkbaar moment in het proces. Dit omdat ASL nogal verschilt met de werkwijze van Thinkwise. Dit maakt dat er voor veel Thinkwise specifieke processtappen ook geen vergelijkbare output te vinden is. Tevens is er, geen output te vinden voor de uitrolfase. Dit komt omdat volgens ASL deze processtap voor de leverancier (Thinkwise in deze context) niet verantwoordelijk is voor de ingebruikname maar dat het enkel ondersteund. (van der Pols, Implementeren, 2009)

Nog een opmerkelijk verschil met het ASL proces is het Testplan als output in de requirements-fase. Dit is al eerder aangehaald in de procesvergelijkingen, maar dit wordt echter ook duidelijk als we inzoomen op de output van een proces. Thinkwise begint met het ontwikkelen van het testplan als het in de lifecycle is aangekomen bij de testfase. ASL stelt dat de testspecificaties al opgesteld moeten worden bij de ontwerpfase en tijdens het opstellen van de requirements. (van der Pols, Ontwerp, 2009)

5.3.12. Deelconclusie

Voor dit hoofdstuk is de volgende deelvraag geformuleerd:

"Wat is de huidige kwaliteit van de gehanteerde software lifecycle?"

In bijlage 3 is de huidige software lifecycle van Thinkwise in kaart gebracht. De lifecycle is opgesteld aan de hand van interviews met leidinggevenden en senior-medewerkers in de praktijk.

Als de Software Development Lifecycle van Thinkwise wordt vergeleken met de best-practices op het gebied van Software Development Lifecycles op een vergelijkbare markt dan zijn er op hoog niveau veel overeenkomsten te zien. De lifecycle van Thinkwise is een mix van de agile- en de iteratieve methode. De juiste methode is gekozen bij de werkwijze in de praktijk en de wijze waarin de lifecycle van Thinkwise verschilt met de best-practices is omdat dit beter aansluit bij de manier van ontwikkelen en de gebruikte programmatuur. Hieruit kan de conclusie getrokken worden dat de knelpunten, welke hebben geleid tot de probleemstelling, zich niet bevinden op hoog-niveau in de lifecycle maar dat er moet worden gekeken naar de onderliggende processen.

De onderliggende processen zijn vergeleken met een best-practice op het gebied van applicatiemanagement: ASL. Na de analyse kan de conclusie getrokken worden dat nagenoeg alle processtappen in de software lifecycle van Thinkwise ook voorkomen in het beheermodel van ASL. Dit is echter niet altijd in hetzelfde procescluster. Dat wil zeggen dat er een probleem ontstaat in de alignment van de organisatie. Dit is bijvoorbeeld het geval bij de Dataconversie-fase. De activiteiten in deze fase zijn in het procesmodel van ASL onderdeel van een andere procescluster. Thinkwise heeft echter deze processtap opgenomen in de ontwikkelprogrammatuur. Het is mogelijk om de activiteiten van Thinkwise behorend bij deze processtap in te delen in hetzelfde procescluster, in het model van ASL, maar dan zou er goed gekeken moeten worden naar de relatie tussen het cluster Programmabeheer en Onderhoud en vernieuwing. Deze processtap is echter zo Thinkwise-specifiek dat het eerst beter onderzocht moet worden voordat hier verbeterstappen op worden geïnitieerd.

Tevens is er, als het raamwerk van ASL als referentiekader wordt gebruikt, weinig aandacht voor de relatie met andere processclusters of dit is slecht inzichtelijk. De verantwoordelijke projectleiders gebruiken vaak hun eigen werkwijze waardoor beheersing vanuit andere processclusters lastig is. De belangrijkste link die hier gemist wordt, is de relatie met de sturende processen, in het bijzonder: Kwaliteitsmanagent. Na elke processtap in ASL is er een terugkoppeling. Dit mist in het ontwikkelproces bij Thinkwise (dit is geconcludeerd in hoofdstuk 5.1.6). Het inzichtelijk krijgen van de relaties van de processtappen met andere processtappen in de lifecycle, of zelfs met andere processclusters kan de alignment en beheersing van de lifecycle verbeteren.

Toch is het een goede zaak dat nagenoeg alle onderliggende processen in de software development lifecycle terug te vinden zijn in vergelijkbare processen in het procescluster van ASL waar de software lifecycle zich bevind. Wijzigingen in de processen en in de alignment zal dan ook moeten plaatsvinden op de punten waar dit niet het geval is.

5.4. Verbeteringen op de software lifecycle

Na het onderzoeken van de huidige kwaliteit van de software lifecycle is het zaak om de geconcludeerde verbeterpunten in kaart te brengen. De resultaten zijn doormiddel van een GAP-analyse gevonden en in kaart gebracht. Hiervoor is de volgende deelvraag geformuleerd:

"Hoe kan de kwaliteit van de software lifecycle verbeterd worden?"

In het vorige hoofdstuk is de kwaliteit van de huidige software lifecycle bij Thinkwise onderzocht. Hier is de conclusie getrokken dat de lifecycle, op zowel hoog niveau als op de onderliggende

processen, aardig voldoet aan kwaliteitsnormen en in veel opzichten overeenkomt met de best-practices in de praktijk. Toch is er weinig inzicht in de kwaliteit van het product. Na de procesanalyse is de conclusie getrokken dat er een knelpunt is ontstaan in de alignment tussen de sturende processen, en in het bijzonder: kwaliteitsmanagement, en de software lifecycle (het ontwikkelproces).

Het proces Kwaliteitsmanagement binnen het procescluster Sturende processen in het beheermodel van ASL heeft als doelstelling dat het zorg draagt voor de kwaliteit van het proces. De relatie met het procescluster waar de lifecycle zich in bevindt is dan ook dat uit alle processen rapportages over de gerealiseerde kwaliteit plaats vinden, aangevuld met evaluaties over de uitvoering van processen en/of eventuele problemen. Om het alignmentprobleem op te lossen moet elk proces een evaluatie geven om kwaliteitsmanagement mogelijk te maken. Het meetbaar maken van het proces is een manier om die evaluatie te realiseren en te documenteren. Een kwaliteitseis moet immers meetbaar zijn. (van der Pols, Kwaliteitsmanagement, 2009)

Allereerst zijn er in samenwerking met verschillende projectleiders en senior-medewerkers de kritische processen en de daarbij behorende metingen opgesteld. Die worden vervolgens vergeleken met de best-practice. Daarnaast wordt er gekeken naar een eventuele verbetering op het procesmatig en op alignment gebied.

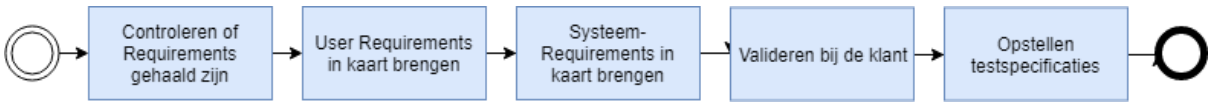
5.4.1. Procesmatige verbeteringen

Uit de analyse in hoofdstuk 5.3 zijn in de vergelijking met ASL een aantal verschillen ontdekt op het niveau van procesinrichting. Op de plekken waar dat nodig is, wordt een proceswijziging gedaan met een onderbouwing uit de literatuur.

Requirements-fase

ASL stelt dat in de requirements-fase al aandacht besteed moet worden aan de manier waarop de requirements worden getest. Een zogenaamd: ‘testontwerp’. Het omvat een beschrijving van de wijze waarop er getest moet worden en de testgevallen waarmee getest moet worden. (van der Pols, Ontwerp, 2009)

Ook is in het ontwikkelproces van ASL overduidelijk te zien dat de functionele specificaties (User requirements) in de processtap vóór het realisatieproces worden opgesteld. In het realisatieproces in ASL worden pas de technische specificaties opgesteld. Bij Thinkwise is dit andersom. Een procesaanpassing zal dus zijn:



FIGUUR 21: PROCESVERBETERING REQUIREMENTS-FASE

Uitrolfase

Zoals ook aangehaald in het vorige hoofdstuk, voert Thinkwise op het moment zelf de acceptatietest uit. ASL stelt dat de afnemer eigenaar van dit proces is. Het proces zal dus niet moeten zijn dat Thinkwise de acceptatietest ontwikkelt, maar de afnemer ondersteund in de uitvoering ervan. (van der Pols, Implementeren, 2009) Tevens is het goed om te benoemen dat het proces: ‘Upgrade-draaiboek opstellen’ als kritisch moet worden gezien. Het nieuwe proces wordt dus:

FIGUUR 22: PROCESVERBETERING UITROLFASE

5.4.2. Alignment verbeteringen

In de analyses op zowel hoog niveau als de onderliggende processen van de software lifecycle is de conclusie getrokken dat de bestaande activiteiten aardig voldoen aan de best-practice. Dit komt doordat de meeste processen zich, in het beheermodel van ASL, zich in het juiste gerelateerde procescluster bevinden; daar waar dus ook de ontwikkeling van de programmatuur plaats zou moeten vinden. Toch loopt de organisatie tegen knelpunten en problemen. Als de onderliggende processen vergelijkbaar zijn aan de best-practice, is de kans dus groot dat deze knelpunten ontstaat in de alignment van de organisatie: De relatie tussen de procesclusters. In dit hoofdstuk worden de resultaten uit de voorgaande analyses gebruikt om tot een advies te komen op het gebied van alignmentverbetering.

In hoofdstuk 5.3.2 wordt al een lichte discussie gevoerd over de activiteiten die bij het referentiekader voor de analysefase horen. Dit is een kwestie van eigenaarschap van het proces. De bepaling of een proces in handen ligt van de leverancier of de afnemer. Aangezien Thinkwise in deze een leveranciersrol vervult, kunnen al veel processen buiten de scope worden geplaatst, aangezien ASL zich ook richt op interne applicatiemanagement en de bijbehorende interne wijzigingen.

Relatie Sturende processen en Onderhoud en vernieuwing

Als in het beheermodel van ASL de relaties van de verschillende processen worden vergeleken met de hantering van de processen bij Thinkwise, is er een structureel verschil te zien in de relatie tussen de processen die geplaatst kunnen worden in het procescluster: Onderhoud en Vernieuwing en het procescluster Sturende Processen. De meest voorkomende inconsistentie zit hier in de relatie met kwaliteitsmanagement. De aansturing van het proces Kwaliteitsmanagement is pas goed mogelijk als het input krijgt uit alle processen in het beheermodel. Dan kan het een eenduidige terugkoppeling geven en tot beter kwaliteitscriteria komen. (van der Pols, Kwaliteitsmanagement, 2009)

Dit houdt in dat voor elk proces in de software lifecycle, welke zich dus in het procescluster van ASL in de cluster Onderhoud en Vernieuwing bevindt, er een terugkoppeling moet worden gedaan naar het kwaliteitssysteem. Aangezien dit natuurlijk geen fysieke afdeling is in het bedrijf gaat het hier dus om het bundelen van deze gegevens en het inzichtelijk krijgen van de resultaten om hier analyses op uit te voeren en kwaliteitsniveaus te bepalen alsmede het tijdig ingrijpen in afwijkingen van de kwaliteitsnormen. (van der Pols, Kwaliteitsmanagement, 2009)

Per fase zal er dus een evaluatie op het proces en de kwaliteitscriteria moeten worden gegeven en zal er vanuit Kwaliteitsmanagement ook een inkomende informatiestroom zijn. Op deze manier wordt de alignment tussen kwaliteitsmanagement en de software lifecycle beter beheersbaar. Hier wordt ASL als referentiekader gebruikt. De relatie met de sturende processen en de software lifecycle ziet er als volgt uit:

<u>Analysefase</u>
Uitgaand: Evaluaties, eventuele problemen, realisatie kwaliteitseisen en indien noodzakelijk eventueel afwijkende aanvullende eisen of wijzigingen in kwaliteitscriteria
Ingaand: kwaliteitscriteria, werkwijzen, indicatoren. (van der Pols, Impactanalyse, 2009)

<u>Requirements-fase</u>
Uitgaand: Eventuele voorgestelde afwijkingen in kwaliteitssysteem, kwaliteitscriteria, etc.
Ingaand: Kwaliteitscriteria, werkwijzen, indicatoren (van der Pols, Ontwerp, 2009)

Designfase
Uitgaand: Eventuele voorgestelde afwijkingen in kwaliteitssysteem, kwaliteitscriteria, etc.
Ingaand: Kwaliteitscriteria, werkwijzen, indicatoren (van der Pols, Ontwerp, 2009)

Ontwikkelfase
Uitgaand: Problemen, evaluaties, eventueel resultaten van audits, collegiale toetsingen, steekproeven.
Ingaand: Eventuele afwijkingen in kwaliteitssysteem, wijze van werken, kwaliteitscriteria, etc. (van der Pols, Realisatie, 2009)

Dataconversie-fase
Uitgaand: Evaluaties, eventuele problemen, realisatie kwaliteitseisen en indien noodzakelijk eventueel afwijkende aanvullende eisen of wijzigingen in kwaliteitscriteria
Ingaand: kwaliteitscriteria, werkwijzen, indicatoren. (van der Pols, Impactanalyse, 2009)

Autorisatiefase
Uitgaand: Evaluaties over de uitvoering van processen en eventuele problemen
Ingaand: Eisen voor wat betreft de continuïteit van de applicatie (van der Pols, Continuïteitsbeheer, 2009)

Testfase
Uitgaand: Evaluaties en eventuele problemen en realisatie kwaliteitseisen. Afwijkende aanvullende eisen, wijzigingen in wijze van werken, kwaliteitscriteria
Ingaand: Kwaliteitscriteria, werkwijzen, indicatoren. (van der Pols, Testen, 2009)

Uitrolfase
Uitgaand: Problemen, evaluaties, eventueel resultaten van audits, collegiale toetsingen, steekproeven.
Ingaand: kwaliteitscriteria, werkwijzen, indicatoren. (van der Pols, Implementeren, 2009)

5.4.3. Kritische processen

Om de vraag te beantwoorden hoe een dashboard kan bijdragen aan de kwaliteit van de software lifecycle is het zaak om kritische processen te definiëren en meetbaar te maken zodat de bovenstaande alignmentverbeteringen kunnen worden toegepast. Op deze manier kan de kwaliteit inzichtelijk worden voor het kwaliteitsmanagement. De kritische processen zijn opgesteld aan de hand van de huidige onderliggende processen in de software lifecycle. De rode processen zijn door de deelnemers van de focusgroep als kritisch in het proces bevonden. Aan de hand van de best-practice ASL wordt de validiteit van deze processen en metingen getoetst.

Analysefase

FIGUUR 23: KRITISCHE PROCESSEN ANALYSEFASE

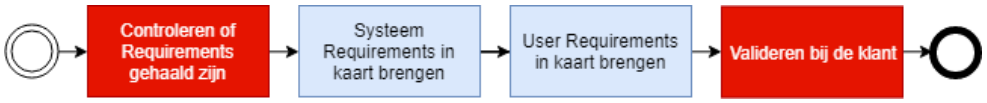
Metingen

- Bevat de projectanalyse alle onderdelen?
- Is de projectanalyse gereviewd door een collega?
- Is de projectanalyse gevalideerd bij de klant?
- Bevat het projectplan alle onderdelen?

Validiteit

In hoofdstuk 5.3.2 is al de validiteit van het proces onderzocht. Het proces voldoet qua processtappen aan de best-practice. Alle processtappen zijn dus nodig in deze fase van de software lifecycle en zouden een goed beeld moeten geven van de volledigheid van dit proces. Het nagaan of deze stappen zijn uitgevoerd is dan ook de meest effectieve meting. Tevens wordt er in ASL benoemd dat de impactanalyse de hoofdlijnen van de gedachte oplossingsrichting bevat. Een basis voor de volgende stap in het proces. (van der Pols, Impactanalyse, 2009).

Requirements-fase



FIGUUR 24: KRITISCHE PROCESSEN REQUIREMENTSFASE

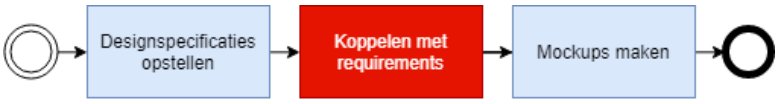
Metingen

- Zijn de Requirements gehaald?
- Zijn de Requirements gevalideerd bij de klant?

Validiteit

In hoofdstuk 5.3.3 is al dieper ingegaan op het proces. Hier is al tot de conclusie gekomen dat volgens ASL in dit proces testspecificaties moeten worden opgesteld. Dit is een beschrijving van de wijze waarop er getest moet worden. Dit is bij Thinkwise niet het geval. Als dit eenmaal als processtap is opgenomen, zal de meting: 'Is de testspecificatie volledig beschreven?' ontstaan en van toegevoegde waarde zijn voor de resultaten verderop in de lifecycle. Verder zijn validatie en accordering van de specificaties een belangrijk onderdeel van de validiteit hiervan. Een belangrijke schakel in de alignment is de voortgang. Deze twee gegevens maken de opgestelde metingen voldoende valide. (van der Pols, Ontwerp, 2009)

Designfase



FIGUUR 25: KRITISCHE PROCESSEN DESIGNFASE

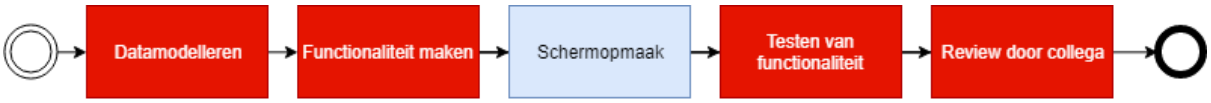
Metingen

- Zijn de designspecificaties gekoppeld met de requirements?

Validiteit

In hoofdstuk 5.3.4 is geconcludeerd dat het koppelen van de designspecificaties met de requirements niet voorkomt in ASL, maar dat het wel een toegevoegde waarde heeft voor de kwaliteit van de alignment en efficiëntie. Aangezien de definitie van een functioneel ontwerp in ASL wordt vergeleken met de User-requirements bij Thinkwise en Thinkwise de designspecificaties enkel gebruikt voor verduidelijking van de oplossingsrichting en werkwijze kan er geconcludeerd worden dat de designspecificaties volgens ASL niet gevalideerd hoeven te worden door de afnemer en enkel onderdeel zijn van de oplossingsrichting. Een extra meting is dus niet nodig. (van der Pols, Ontwerp, 2009)

Ontwikkelfase



FIGUUR 26: KRITISCHE PROCESSEN ONTWIKKELFASE

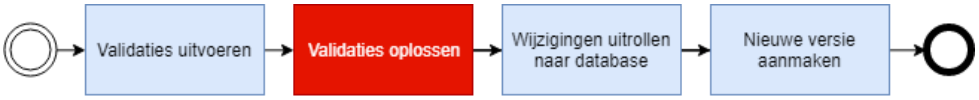
Metingen

- Zijn er fouten gevonden in het datamodel?
- Zijn er fouten gevonden in de functionaliteit?
- Zijn alle functionaliteiten (codes) getest?
- Zijn alle functionaliteiten gereviewd door een collega?

Validiteit

Zoals besproken in hoofdstuk 5.3.5 wordt er in ASL niet gespecificeerd hoe het ontwikkelproces in de ontwikkelfase eruit ziet. Dit is namelijk afhankelijk van de ontwikkelmethodiek of programmatuur waarmee de software wordt ontwikkeld. (van der Pols, Realisatie, 2009) Het is dus ook lastig om de validiteit van de metingen te valideren. Om toch een vergelijking te maken worden de processen: datamodelleren, functionaliteit maken en schermopmaak met elkaar gecombineerd onder de noemer; realiseren. ASL stelt dat de programmatuur wordt gerealiseerd en vervolgens wordt getest. De metingen van het realiseren is het wegnemen van fouten en de metingen bij het testen en de review de controle of het is uitgevoerd.

Dataconversie-fase



FIGUUR 27: KRITISCHE PROCESSEN DATACONVERSIE-FASE

Metingen

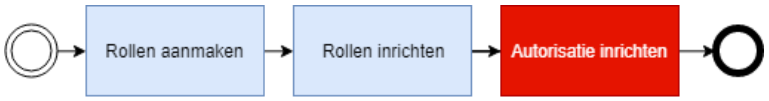
- Zijn alle validaties opgelost?

Validiteit

Een gegeven wat al vaker is aangehaald is het feit dat de Thinkwise ontwikkelprocessen afwijken van de ‘standaard’. Uit de focusgroep is in dit proces het enige kritische proces; het oplossen van validaties, aangewezen. Dit is op zichzelf een uitstekende meting omdat het gaat om het oplossen van de problemen in die set van wijzigingen. Die validaties uitvoeren worden in ASL beschreven als: het in kaart brengen wijziging, het inschatten van veranderingen, en het inschatten van consequenties. Dit zijn volgens ASL de belangrijkste indicatoren voor de continuïteit en de kwaliteit van de applicatie. (van der Pols, Impactanalyse, 2009)

Wat echter wel als kritisch proces mist is het aanmaken van een nieuwe versie. Dit is van essentieel belang aangezien ASL stelt dat versiebeheer een veilige werkwijze moet bieden die de risico’s van ongeautoriseerd gebruik, ongeautoriseerde wijzingen en vernietiging moeten beperken. (van der Pols, Programmabeheer en distributie, 2009). Een voorgestelde meting hierbij is dan ook: Is er een nieuwe versie aangemaakt na de vorige databasewijziging?

Autorisatie



FIGUUR 28: KRITISCHE PROCESSEN AUTORISATIEFASE

Meting

- Zijn alle rollen volledig ingericht?

Validiteit

Uit de analyse in hoofdstuk 5.3.7 is de conclusie getrokken dat volgens ASL de afnemer verantwoordelijk is voor de bepaling van de autorisatie van de applicatie. Hiervoor hoeft er geen meting plaats te vinden. De enige controle die de applicatie hier op uit kan voeren is het controleren of de input met betrekking tot de autorisatie volledig is uitgevoerd en verwerkt. De mate van volledigheid hangt dan af van de input van de afnemer. (van der Pols, Continuïteitsbeheer, 2009)

Testen

FIGUUR 29: KRITISCHE PROCESSEN TESTEN

Metingen

- Zijn alle functionaliteiten gedekt met een unittest?
- Zijn alle unittests succesvol afgerond?

Validiteit

De unittest wordt vaak aangehaald in ASL. Het uitvoeren van de unittests zijn dan ook van essentieel belang om binnen het ontwikkelproces te controleren of het gemaakte product nog voldoet aan het ontwerp. Daarnaast wordt er tevens door een persoon getest. Dit is echter niet benoemd in ASL. Het niet kiezen voor een kritisch proces is hier dan ook de juiste keuze. (van der Pols, Testen, 2009)

Uitrollen

FIGUUR 30: KRITISCHE PROCESSEN UITROLFASE

Metingen

- Is de acceptatietest volledig uitgevoerd?
- Bevat het deploymentplan alle onderdelen?
- Is alles klaar voor productie?

Validiteit

In hoofdstuk 5.3.9 is de validiteit van de onderliggende processen onderzocht. Hier is de conclusie uitgekomen dat, volgens ASL, de acceptatietest niet door de leverancier ontwikkeld moet worden. Het opstellen ervan is dan ook geen kritisch proces. Het uitvoeren ervan echter wel. Dit is dan ook in de meting terug te vinden. Wat opvalt is dat er in de focusgroep is benoemd is dat er een draaiboek wordt bijgehouden van de upgrade. Dit fungeert als het documenteren van de wijzigingen in het raamwerk van ASL. Om die reden kan het als kritisch worden beschouwd omdat het essentieel is voor overname

van het beheer van de applicatie. (van der Pols, Implementeren, 2009) Een toegevoegde meting kan dan zijn; ‘Bevat het Upgrade-draaiboek alle onderdelen?’.

5.4.4. Deelconclusie

Voor dit hoofdstuk is de volgende deelvraag geformuleerd:

“Hoe kan de kwaliteit van de software lifecycle verbeterd worden?”

Om de probleemstelling op te lossen moet er gekeken worden naar de beheersing van de software lifecycle. In hoofdstuk 5.3, waar ASL als referentiemodel is gebruikt, is geconcludeerd dat het knelpunt is ontstaan bij de alignment van het procescluster waar de lifecycle zich in bevindt en het procescluster waar zich kwaliteitsmanagement bevindt. Het is dan ook niet meer dan logisch dat er in de praktijk problemen ontstaan bij de inzicht in de algemene kwaliteit van de applicatie.

Om dit beheersbaar te krijgen moet er een betere relatie ontstaan tussen de software development lifecycle van Thinkwise en de aansturing van kwaliteitsmanagement. Dit is aangepakt in 3 stappen.

Allereerst is er onderzocht op welke punten in de lifecycle procesverbetering plaats moet vinden. In de analyse uit hoofdstuk 5.3 is er al een procesanalyse gedaan. Hier is de conclusie getrokken dat de invulling van de processen over het algemeen in de goede procesclusters plaats vinden en, op een enkele na, alle activiteiten aanwezig zijn in ASL in vergelijkbare processtappen. De benodigde aanpassingen; Het opstellen van testspecificaties in de requirements-fase en het ondersteunen bij de acceptatietest in plaats van het ontwikkelen en uitvoeren ervan, zijn van dusdanig kleine aard dat de proceswijziging weinig impact heeft op de rest van de lifecycle.

Vervolgens is de alignment onder de loep genomen. In hoofdstuk 5.1.6 is hier een duidelijke, en tevens meest belangrijke, conclusie getrokken dat er geen terugkoppeling wordt gegeven na elke processtap in de software lifecycle. Dit wil zeggen dat de alignment tussen de sturende processen op het gebied van kwaliteitsmanagement en de uitvoering van de lifecycle moeilijk beheersbaar is. Er is geen input (evaluatie) voor kwaliteitsmanagement op de lifecycle en dat maakt dat het ook geen output (kwaliteitscriteria) terug kan geven.

ASL stelt dat door middel van een constante evaluatie van de gegevens, werkwijze, resultaten en andere relevante kwaliteitsaspecten na elke processtap te doen, er een inzicht komt in de kwaliteit van het proces en de applicatie. Dit maakt het proces beheersbaar en kan de terugkoppeling in de vorm van eisen, werkwijzen en kwaliteitscriteria worden gedaan. In hoofdstuk 5.4.2 wordt uitgelegd wat de relatie zal worden tussen de software lifecycle en het kwaliteitsmanagement bij Thinkwise.

Aan de hand van de bovenstaande conclusies is het proces weer onder de loep genomen. Dit keer om de kritische processen te bepalen en de metingen op te stellen die moeten zorgen voor de relatie tussen de software lifecycle en het kwaliteitsmanagement. De metingen zijn getoetst op ASL en eventueel aangevuld op de punten waar dit nodig is.

Door de metingen na elke fase in de software lifecycle uit te voeren wordt de relatie met kwaliteitsmanagement mogelijk en kan er een beeld worden geschetst van de algemene kwaliteit van de applicatie. Een directe oplossing voor de probleemstelling.

5.5. Implementatie van de veranderingen

Nu de verbeteringen zijn geïdentificeerd en in kaart zijn gebracht is de resterende vraag hoe dit in de organisatie geïmplementeerd kan worden. De deelvraag die hierbij is geformuleerd luidt:

“Hoe moeten de benodigde verbeteringen worden geïmplementeerd?”

Om te onderzoeken welke stappen er nodig zijn om de oplossing te implementeren wordt er in dit hoofdstuk tevens onderscheid gemaakt tussen procesmatige verbeteringen en de verbeteringen op het gebied van alignment. Het resultaat van dit hoofdstuk is een stappenplan om deze aanpassingen in de organisatie te implementeren.

5.5.1. Implementatie van de procesverbeteringen

De veranderingen op procesmatig gebied zijn in de omvang nihil. Het gaat hierbij om het opstellen van de user-requirements voordat de systeem-requirements worden opgesteld en het ondersteunen van de acceptatietest in plaats van het ontwikkelen hiervan. Om te zorgen voor een goede doorvoering en de beheersing van de aanpassing worden de benodigde stappen beschreven, wordt er beschreven hoe de medewerkers de verandering positief kunnen beïnvloeden en hoe de verandering kan worden gemeten om te valideren dat dit daadwerkelijk van toegevoegde waarde is.

Doorvoering procesverandering

Wegens de kleine omvang van de in hoofdstuk 5.4.1 (Procesmatige verbeteringen) beschreven veranderingen is het niet nodig om tot grote veranderstrategieën te komen. De impact is hierbij ook te klein om dit door middel van een volwassen methode door te voeren of de veranderingen door middel van een compleet verandermodel de organisatie in te slingeren. Dit zou niet efficiënt zijn, aangezien het uitvoeren van dergelijke verandermodellen gericht zijn op het wijzigen van grote, kritieke en veelomvattende bedrijfsprocessen. Thinkwise hanteert zijn eigen projectmethodiek. Deze methodiek wordt geregeld aangepast aan de hand van veranderingen in de markt of naar aanleiding van een analyse van het management. In de projectmethodiek worden beide aspecten, zowel het opstellen van requirements als het opstellen van het acceptatieplan, beschreven. (Thinkwise B.V., 2017) Het opnemen van deze verbeteringen in een toekomstige versie van de projectmethodiek is daarbij voldoende voor de implementatie van de procesveranderingen.

Eigenaarschap

Uit interviews is gebleken dat projectleiders regelmatig afwijken van de Thinkwise projectmethodiek. Dit heeft te maken met het feit dat zij, vanwege hun expertise, een andere werkwijze efficiënter vinden en ervoor kiezen deze uit te voeren. In dit onderzoek is ook geconcludeerd dat deze projectleiders eigenaar zijn van het proces: Kwaliteitsmanagement. Dat wil zeggen dat zij ook verantwoordelijk zijn voor wijziging in de werkwijze als dit noodzakelijk is voor het behalen van een hogere mate van kwaliteit. Om de procesveranderingen door te voeren en er tegelijkertijd voor te zorgen dat dit ook in de praktijk wordt gehanteerd moeten medewerkers worden betrokken bij de verandering zodat zij zich eigenaar voelen van de door te voeren veranderingen waardoor de weerstand wordt weggenomen.

Het toepassen van het WWK-model (behandeld in hoofdstuk 3.6.1) kan de weerstand op deze veranderingen wegnemen. Om het in de context van dit onderzoek te plaatsen:

- De projectleiders zo goed mogelijk infomeren over de door te voeren procesverbetering, alle informatie verschaffen over de reden van verandering en uitleggen wat dit betekent voor de toekomstige werkwijze (weten).
- De projectleiders vertrouwen geven in succesvolle uitvoering van de projecten, betrokken maken in volgende versies van projectmethodieken, actief mee laten denken in het ontwikkelen van de Thinkwise best-practice (willen)
- Positief benaderen in de uitvoering van de processen, kennisdeling onderling stimuleren en middelen toewijzen die benodigd zijn voor succesvolle uitvoering hiervan.

Beheersbaarheid
Om de veranderingen te toetsen en te weten of het in de toekomst ook efficiënt blijft, is het goed om deze te blijven meten. Dit is mogelijk als er regelmatig een evaluatie van het proces plaats vind. (van der Pols, Kwaliteitsmanagement, 2009) Op het moment dat er uit die analyse de conclusie wordt getrokken dat deze veranderingen niet de gewenste kwaliteit oplevert, zal er een nieuwe inrichting van het proces plaats moeten vinden.

5.5.2. Implementatie van de alignmentverbeteringen

Uit de analyse in hoofdstuk 5.4 (Verbeteringen op de software lifecycle) is tevens een verbeterpunt op het gebied van alignment vastgesteld. Het gaat hier om het evalueren van elke fase van de software lifecycle. Om dit mogelijk te maken zijn er metingen opgesteld om de resultaten van een proces te meten en zo in een fase een aantal procesmetingen te doen. Het onderzoeken van de implementatiemogelijkheden van deze verbeteringen kijken we naar de ontwikkelsoftware van Thinkwise en de mogelijkheid tot integratie van de nieuwe functionaliteiten op de huidige programmatuur. Dit wordt onderzocht door te kijken welke metingen geautomatiseerd kunnen worden door de SF, door welke wijze de informatie vergaard kan worden, en of dit al mogelijk is in de huidige SF. Vervolgens wordt er een schatting gemaakt van de uren die nodig zijn voor het ontwikkelen van het dashboard.

Integratie met SF
Thinkwise heeft met het gebruik van de SF een uitstekend product voor het realiseren van een goede softwarematige ondersteuning op bedrijfsprocessen. Aangezien zij software ontwikkelen met een eigen ontwikkelde programmatuur, geeft het hen de flexibiliteit om de gebruikte programmatuur op een dusdanige wijze aan te passen dat het ook de eigen bedrijfsprocessen zo goed mogelijk ondersteund. (Thinkwise Software B.V., 2019)

In tabel 9 is er aan de hand van een interview met Kevin Horst (manager Software Modernization) vastgesteld hoe de metingen in de SF kunnen worden geïmplementeerd. De wijze waarop de SF is opgezet maakt het mogelijk om veel metingen te automatiseren. Er hoeft dan geen menselijke handeling aan te pas komen omdat de metingen kunnen worden gedaan door andere functionaliteiten of geregistreerde data in de software. Dit is volgens Kevin Horst het meest wenselijke geval. Het geeft het meest betrouwbare en volledige resultaat.

Wat opvalt in Tabel 9 is dat nog niet elke meting in de huidige versie van de SF mogelijk is. Dit is vooral gericht op de documentatie. Kevin Horst raadt aan om dit, om zo volledige beheersing te behouden, in de SF uit te voeren. Dit is volgens hem echter niet altijd mogelijk. Uit interviews met projectleiders is gebleken dat dit in de praktijk gebeurt door het opstellen van een word-document. Om dit te managen, meetbaar te maken en te evalueren, is het volgens Kevin Horst zaak om de uitvoering van deze documenten in een project in de vorm van een checklist te evalueren. De volledigheid van deze checklist is dan het resultaat van de meting. Voor de realisatie van het dashboard zullen dus eerst deze checklist ontwikkeld moeten worden. Als het in de toekomst mogelijk is om de documentatie van zowel de analyse-fase als de uitrol-fase in de software op te stellen, kan de meting worden geautomatiseerd. Dit is de meest wenselijke situatie.

Ten slotte bestaat er in Tabel 9 ook een derde wijze van meten: Controle. Dit zijn vaak review functies. Dat wil zeggen dat een gebruiker enkel hoeft aan te geven of een bepaalde handeling is uitgevoerd of niet. Dit kan, volgens Horst, in de SF doormiddel van een taak of een extra kolom in de database. De waarde van die kolom is dan het resultaat van de meting.

Kosten

Uit het interview over de integratie van de metingen is gebleken dat alle metingen kunnen worden gerealiseerd in de huidige programmatuur van Thinkwise. Dat wil zeggen dat er geen externe applicatie moet worden geïntegreerd om de veranderingen te implementeren. Het ontwikkelen van het dashboard zal dus intern kunnen worden uitgevoerd. De hierbij behorende kosten zitten hier dus in de loonkosten van een werknemer. In Tabel 10 is een schatting gemaakt van de realisatie van het dashboard. De totale geschatte uren voor de realisatie van het dashboard komt op 76 uur. De schatting is gemaakt door Kevin Horst en geverifieerd door Robert van Tongeren en Robert-Jan de Nie.

5.5.3. Stappenplan implementatie

Naar aanleiding van de bovenstaande analyse is er een stappenplan opgesteld voor de implementatie van de oplossing. Deze is te vinden in bijlage 14. Hierbij is er in Figuur 15 ook een visuele weergave gemaakt van het te doorlopen proces. Wat hierbij opvalt is het feit dat de ontwikkeling van het dashboard plaats vindt voordat de procesmatige veranderingen worden doorgevoerd. Dit komt omdat het dashboard en de daarbij behorende werkwijze tevens moet worden opgenomen in de projectmethodiek omdat dit een belangrijke tool wordt van de kwaliteitsmanagement in een project. In Figuur 31 zijn er tevens 3 ‘lanes’ te zien. Dit zijn drie verschillende rollen die worden betrokken bij de implementatie. Het gaat om de volgende rollen:

- **De ontwikkelaar van het dashboard**
  Dit kan iedere ontwikkelaar binnen Thinkwise zijn en het is dus niet noodzakelijk dat hij/zij een specifieke, verantwoordelijke functie heeft.
- **De directie (board)**
  Ten tweede is de directie gespecificeerd omdat deze verantwoordelijk is voor de ontwikkeling en doorvoering van de Thinkwise projectmethodiek.
- **Verantwoordelijken**
  Deze zijn verantwoordelijk voor het meedenken in het ontwikkelen van een nieuwe of verbeterde projectmethodiek en het opstellen van de kwalificatiecriteria in een project.

5.5.4. Deelconclusie

Voor dit hoofdstuk is de volgende deelvraag geformuleerd:

“Hoe moeten de benodigde verbeteringen worden geïmplementeerd?”

Aan de hand van de resultaten van de voorgaande deelvragen zijn er verschillende verbeteringen opgesteld. Deze verbeteringen kunnen worden verdeeld in procesmatige- en alignmentverbeteringen. De procesmatige verbeteringen zijn van dusdanig kleine impact dat deze gemakkelijk kunnen worden opgenomen in een toekomstige versie van de Thinkwise projectmethodiek. Aangezien er uit analyse is gebleken dat projectleiders van Thinkwise geregeld afwijken van de gehanteerde methodiek, is het goed om medewerkers te betrekken bij het opstellen van toekomstige projectmethodieken om zo gedeelde eigenaarschap te behalen en de weerstand op de veranderingen te voorkomen.

Om de alignmentverbeteringen te implementeren is er naast een procesmatige aanpassing ook een aanpassing nodig op de software. Uit interviews is gebleken dat alle opgestelde metingen kunnen worden uitgevoerd door de huidige programmatuur. De realisatie van het dashboard zal dus intern kunnen worden uitgevoerd en de bijbehorende kosten zullen dus zijn in de vorm van loonkosten. De geschatte aantal uren zijn vastgesteld op 76 uur voor totale realisatie van het dashboard. Om tot realisatie te komen is er een stappenplan gerealiseerd. Het stappenplan in bijlage 14 heeft als doel om te specificeren welke stappen er nodig zijn om het dashboard te realiseren en de overige wijzigingen door te voeren.

6. CONCLUSIE

De probleemstelling van het onderzoek is voorafgaand aan het onderzoek vastgelegd. Hierin geeft Thinkwise aan dat het in de huidige situatie tijdens het ontwikkelproces moeilijk in staat is om inzicht te krijgen in de algemene kwaliteit van de applicatie. Per behandeld hoofdstuk worden hier de deelconclusies nogmaals herhaald en wordt er vervolgens een eindconclusie getrokken.

6.1. Wat wordt er verstaan onder kwaliteit en hoe wordt dit beheerst in een organisatie?

Naar aanleiding van het literatuuronderzoek voor de definitiebepaling voor kwaliteit is de volgende definitie opgesteld:

"De kwaliteit van de software lifecycle is de mate waarin de software lifecycle voldoet aan de eisen van het kwaliteitsmanagement van de organisatie binnen de gewenste tijdkaders"

In het kwaliteitsmanagement van de organisatie worden de eisen van de lifecycle bepaald.
De definitie van kwaliteitsmanagement die hierbij aangehouden wordt is: "*Alle activiteiten binnen het totale management-orgaan die die kwaliteitseisen, doelen en verantwoordelijkheden bepalen en deze implementeren door kwaliteits-planning, -controle en -verbeteringen toe te passen*"

In het kwaliteitsmanagement wordt zowel de kwaliteit van het eindproduct als de kwaliteit van het proces beheerst. De verwachting en de eisen van de afnemer zijn dus onderdeel van de eisen voor de lifecycle die door het kwaliteitsmanagement van de organisatie worden opgesteld. De tijdkaders wordt per project bepaald in overeenstemming met de klant.
Als er wordt gekeken naar kwaliteitsmanagement bij Thinkwise zijn er twee zaken die opvallend verschillen met de best-practice ASL. De mate en wijze waarin er evaluaties plaats vinden in het project en de wijze waarin afwijkingen van de kwaliteit worden teruggekoppeld naar de afnemer. Er is een correlatie te vinden tussen deze twee ontwikkelingen in het proces. Als de projectleiders tijdens het ontwikkelproces het overzicht in de kwaliteit verliezen komt het logischerwijs bij de afronding van het project tot verassingen te staan. Om deze verassingen te voorkomen zal Thinkwise tijdens de uitvoering van het ontwikkelproces meer inzicht moeten verschaffen in de ontwikkelingen en tijdens elke processtap een evaluatiemoment uit moeten voeren. Dit creëert een overzicht voor projectleiders in de algemene kwaliteit van het eindproduct waardoor ze sneller in kunnen spelen op afwijkingen van de eisen en kunnen deze afwijkingen sneller met de afnemer worden gecommuniceerd om de aangepaste verwachtingen helder te krijgen.

6.2. Wat maakt een software lifecycle succesvol?

De Software Lifecycle, ook wel Software Development Life Cycle (verkort: SDLC) is een workflow proces welke de kernfasen en activiteiten in een ontwikkelcyclus definiëren. Het is gericht op het ontwikkelen van software met de hoogst mogelijke kwaliteit en de laatste kosten in de kortste tijd. Het omvat een gedetailleerd plan voor het ontwikkelen, aanpassen, onderhouden en vervangen van een informatiesysteem.
Een Software Development Lifecycle is altijd afhankelijk van de ontwikkelmethode en werkwijze van een organisatie. De toepassing ervan moet dan ook passen bij de verschillende aspecten die het ontwikkelproces definiëren. Dit omvat zowel de mate waarin de requirements vooraf worden verduidelijkt, de communicatie met en de betrokkenheid van de klant, de omvang van het project en de ontwikkelmethodiek. Een succesvolle software lifecycle is dus op een dusdanige wijze ingericht dat het inspeelt op de afspraken binnen het project en dit aansluit bij de methode waarop er software wordt ontwikkeld.

6.3. Wat is de huidige kwaliteit van de gehanteerde Software Lifecycle?

In bijlage 3 is de huidige software lifecycle van Thinkwise in kaart gebracht. Als de Software Development Lifecycle van Thinkwise wordt vergeleken met de best-practices op het gebied van Software Development Lifecycles dan zijn er op hoog niveau veel overeenkomsten te zien. Hieruit kan de conclusie getrokken worden dat de knelpunten, welke hebben geleid tot de probleemstelling, zich niet bevinden op hoog-niveau in de lifecycle maar dat er moet worden gekeken naar de onderliggende processen. De onderliggende processen zijn vergeleken met een best-practice op het gebied van applicatiemanagement: ASL. Uit analyse blijkt dat nagenoeg alle onderliggende processen in de software development lifecycle terug te vinden zijn in vergelijkbare processen in het procescluster van ASL waar de software lifecycle zich bevind. Het knelpunt zit dus niet in de hantering van de procesfasen zelf maar in de relatie tot andere processen; de alignment. De belangrijkste link die hier gemist wordt, is de relatie met de sturende processen, in het bijzonder: Kwaliteitsmanagent. Na elke processtap in ASL is er een terugkoppeling. Dit mist in het ontwikkelproces bij Thinkwise. Het inzichtelijk krijgen van de relaties van de processtappen met andere processtappen in de lifecycle, of zelfs met andere procesclusters kan de alignment en beheersing van de lifecycle verbeteren.

6.4. Hoe kan de kwaliteit van de software lifecycle verbeterd worden?

ASL stelt dat door middel van een constante evaluatie van de gegevens, werkwijze, resultaten en andere relevante kwaliteitsaspecten na elke processtap te doen, er een inzicht komt in de kwaliteit van het proces en de applicatie. Dit maakt het proces beheersbaar en kan de terugkoppeling in de vorm van eisen, werkwijzen en kwaliteitscriteria worden gedaan. In een focusgroep zijn daarom de kritische processen van de Software Lifecycle bepaald en zijn er metingen opgesteld die moeten zorgen voor de relatie tussen de software lifecycle en het kwaliteitsmanagement.

6.5. Hoe moeten de benodigde verbeteringen worden geïmplementeerd?

Aan de hand van de resultaten van de voorgaande deelvragen zijn er verschillende verbeteringen opgesteld. Deze verbeteringen kunnen worden verdeeld in procesmatige- en alignmentverbeteringen. De procesmatige verbeteringen zijn van dusdanig kleine impact dat deze gemakkelijk kunnen worden opgenomen in een toekomstige versie van de Thinkwise projectmethodiek. Hierbij is het belangrijk om medewerkers te betrekken bij het opstellen van toekomstige projectmethodieken om zo gedeelde eigenaarschap te behalen en de weerstand op de veranderingen te voorkomen.

Om de alignmentverbeteringen te implementeren is er naast een procesmatige aanpassing ook een aanpassing nodig op de software. Uit interviews is gebleken dat alle opgestelde metingen kunnen worden uitgevoerd door de huidige programmatuur. De geschatte aantal uren zijn vastgesteld op 76 uur voor totale realisatie van het dashboard. Om tot realisatie te komen is er een implementatieplan gerealiseerd

6.6. Eindconclusie

Voorafgaand aan dit onderzoek is aan de hand van de probleemstelling de volgende hoofdvraag geformuleerd:

“Hoe kan Thinkwise de kwaliteit van haar software lifecycle op een duurzame wijze verbeteren?”

Uit dit onderzoek is gebleken dat het ontbreken van de inzicht in de kwaliteit van de applicatie een gevolg is van een knelpunt in de alignment van het bedrijf. Om dit te verbeteren is het beheermodel ASL als referentiekader gebruikt. Het belangrijkste knelpunt is te vinden in de relatie tussen de uitvoering van de software lifecycle, welke zich in het procescluster Onderhoud en Vernieuwing van het beheermodel van ASL bevind, en de kwaliteitsmanagement, welke zich in het procescluster

Sturende processen bevind. ASL stelt dat na elke processtap een constante evaluatie van de gegevens, werkwijze, resultaten en andere relevante kwaliteitsaspecten te doen, er een inzicht komt in de kwaliteit van het proces en de applicatie. Dit maakt het proces beheersbaar en kan de terugkoppeling in de vorm van eisen, werkwijzen en kwaliteitscriteria worden gedaan. Het meetbaar maken van dit proces zorgt dat er output wordt gecreëerd. Door de metingen na elke fase in de software lifecycle uit te voeren wordt de relatie met kwaliteitsmanagement mogelijk en kan er een beeld worden geschetst van de algemene kwaliteit van de applicatie.

Aan de hand van deze conclusie is er naar aanleiding van dit onderzoek een stappenplan opgesteld om de in dit onderzoek vastgestelde metingen in de organisatie te implementeren. De processen en activiteiten behorend bij kwaliteitsmanagement moeten met deze input door middel van een terugkoppeling in de vorm van eisen, werkwijzen en kwaliteitscriteria zorgen voor een betere kwaliteit van het gehanteerde ontwikkelproces en dus van de gehanteerde software lifecycle.

7. AANBEVELINGEN

In dit hoofdstuk worden aanbevelingen gegeven die op basis van de resultaten en conclusies zijn gemaakt. De aanbevelingen zijn gebaseerd op observaties tijdens interviews, conclusies naar aanleiding van de vergelijking van de praktijk met de literatuur en de beantwoording van de hoofd- en deelvragen. De aanbevelingen aan de hand van dit onderzoek zijn:

- User-requirements opstellen voordat de systeem-requirements worden opgesteld en het opstellen van testspecificaties in de requirements-fase. Door de volgorde van requirements in het ontwikkelproces van Thinkwise om te draaien kunnen er kwalitatief betere systeem-requirements worden opgesteld en de wijze van testen gerichter worden beschreven.
- De klant moet zelf verantwoordelijk gemaakt worden voor het uitvoeren van de acceptatietest op de software van Thinkwise in haar eigen omgeving. Op dit moment ontwikkelt Thinkwise deze zelf en voert deze ook uit. Dit maakt dat de validatie van de test in twijfel wordt getrokken omdat de test als het ware 'gemanipuleerd' wordt. Het uitvoeren van de acceptatietest door de klant zelf, met Thinkwise als ondersteunende factor, zorgt voor meer realistischere testresultaten.
- Op het moment worden sommige documenten nog in aparte programma's opgesteld. Om alle processen in de SF meetbaar te maken, en een duidelijk overzicht te creëren, moeten alle processtappen ook in de SF worden uitgevoerd of moet er een koppeling met de SF bestaan. Door de documentatie in de software op te nemen, kan de evaluatie (en dus de meting) hierop geautomatiseerd worden. Dit kan ook handmatig met een checklist.
- De metingen die zijn opgesteld in de resultaten zullen in een dashboard op een centrale plek in de SF moeten worden opgenomen om zo een duidelijk beeld te schetsen van de algemene kwaliteit van het eindproduct in een project. Op deze wijze kan de projectverantwoordelijke het dashboard gebruiken als hulpmiddel voor de kwaliteitsmanagement van zijn project en fungeert deze als evaluatiemiddel van de gehanteerde processen.
- Om de alignmentverbeteringen op duurzame wijze te blijven verbeteren is het goed om het proces en de kwaliteitsmanagement in een project te blijven evalueren. Merkt een projectleider dat het, op welke wijze dan ook, te weinig inzicht heeft op de uitvoering van een proces, dan zullen er metingen ontwikkeld moeten worden om de missende componenten toe te voegen aan het dashboard. Tevens zijn wijzigingen in de projectmethodiek, wijzigingen in de SF, wijzigingen in de werkwijze en nieuw gebruikte technologieën redenen om het dashboard op volledigheid te evalueren.
- De uitvoerende projectleider wijkt in de praktijk geregeld af van de projectmethodiek van Thinkwise. Door de medewerkers actief te laten participeren in het ontwikkelen van de gehanteerde methodiek, wordt er eigenaarschap gecreëerd. Vraag hierbij projectleiders en senior medewerkers om input, organiseer focusgroepen en organiseer brainstormsessies om een gezamenlijk gehanteerde en gedragen methodiek te ontwikkelen.
- Toets de Thinkwise projectmethodiek met best-practices als ASL, ITIL of andere beheermodellen die veel worden gebruikt in een vergelijkbare markt. Dit is op dit moment niet het geval. Door te onderzoeken waar er verschil plaats vindt tussen de projectmethodiek van Thinkwise en de best-practices kan er een betere methodiek worden ontwikkelt.
- In dit onderzoek is enkel een deel van de best-practice ASL gebruikt. Toch kan ASL als referentiemiddel voor de gehele organisatie worden ingezet en niet enkel op de uitvoering van projecten. Het is dan ook aanbevelen om te onderzoeken wat ASL nog meer kan opleveren op andere 'clusters' binnen de organisatie die niet zijn meegenomen in de scope van dit onderzoek.

BRONNENLIJST

Bakker, R., & Spronk, W. (2011). *Het Procesmanagement Modellenboek.* Deventer: Vakmedianet Management B.V.

Balaji, S., & Sundararajan Murugaiyan, M. (2012). *Waterfall vs V-model vs Agile: A comparative study on SDLC.* Chennai, TN, India: JITBM & ARF.

Brom, D. (2019). *Plan van Aanpak.* Apeldoorn: Saxion Deventer.

CBO. (2004). *Handleiding focusgroepen/kwaliteitsinstituut voor de gezondheidszorg CBO.* CBO.

De Win, B. (2017). SDLC Maturity Models. *SecAppDev 2017*, 5. PWC.

Deming, W. E. (2000). *Out of this Crisis.* Cambridge, Massachusetts, VS: MIT Press.

Department of Justice U.S.A. (2003). *Chapter 1: Introduction to System Development Life Cycle (SDLC).* Justice Management Division.

Garvin, D. A. (1984). Wat does "Product Quality" really mean? *Sloan Management Review*, 19.

Hackman, R., & Wageman, R. (1995). *Total Quality Management: Empirical Conceptual en Practical Issues.* Cornell: Sage Publications.

Hogeschool Rotterdam. (2014). *Handleiding zoekstrategie deskresearch.* Rotterdam.

Horst, K. (2019, Juni 12). Integratie metingen in SF. (D. Brom, Interviewer)

Hoving, W., & van Bon, J. (2013). *De ISM-Methode Versie 3 - Verleden heden en toekomst van IT-servicemanagement.* Academic Service, een imprint van SDU Uitgevers.

Imam, A. (2018, Augustus 27). *Software Development Life Cycle: The phases of SDLC*. Opgehaald van Testlodge: https://blog.testlodge.com/software-development-life-cycle/

INK. (2019, 03 18). *INK-managementmodel*. Opgehaald van INK: https://www.ink.nl/model/ink-managementmodel/

Institute of Technology Tallaght. (2007). *Quality management and quality assurance standards - BS EN ISO 9000-1:1994.* Tallaght: BSI.

Juran, J. M. (1998). How to think about quality. In J. M. Juran, & A. B. Godfrey, *Juran's Quality Handbook* (pp. 20-21). New York: McGraw Hill Companies Inc.

Kloost, J. (2019, April 8). Technische aftrap afstudeeronderzoek. (D. Brom, Interviewer)

leansixsigmatools. (2011, December 14). *5 x waarom methode*. Opgehaald van Leansixsigmatools: https://leansixsigmatools.nl/2011/12/14/5-waarom-methode-5-whys

Looijen, M., & Delen, G. (1992). *Beheer van informatievoorziening.* Rijswijk: Cap Gemini Publishing.

Looy, E. (2011). *Defining business process maturity; A Journey towards Excellence.* Total Quality Management.

Lucidchart. (2019, Februari 21). *Wat is Business Process Modeling Notation?* Opgehaald van Lucidchart: https://www.lucidchart.com/pages/nl/wat-is-business-process-modeling-notation?a=0

Marquardt, D. (1998). The ISO 9000 Family of International Standards. In J. Juran, & A. Godfrey, *Juran's Quality Handbook* (p. 3). New York: McGraw-Hill Companies, Inc.

Nooij, W. (2015, Aug 26). *Hoe voorkom je weerstand bij verandering?* Opgehaald van Logistiek: https://www.logistiek.nl/carriere-mensen/blog/2015/08/merk-jij-ook-weerstand-als-je-de-

klant-wilt-verrassen-101135628?vakmedianet-approve-cookies=1&_ga=2.23819002.1951561942.1560255025-779814527.1560255025

Parzinger, M., & Nath, R. (2000). *A Study of the relationships between total quality management implementation factors and software quality.* California Plaza, Omaha, USA: Carfax Publishing.

Paulk, M., Curtis, B., Chrissis, M., & Weber, C. (1993). *Capability Maturity Model for Software, Version 1.1.* Pittsburgh: Software Engineering Institute.

Pinheiro, J. (2018, April 11). *Software Developtment Life Cycle (SDLC) phases*. Opgehaald van Medium: https://medium.com/@jilvanpinheiro/software-development-life-cycle-sdlc-phases-40d46afbe384

Pyzdek, T., & Keller, P. (2010). *The Six Sigma Handbook.* New York: The McGraw-Hill Companies, Inc.

Röglinger, M. (2012). *Maturity models in business process management; Business Process management journal.*

Sigmaonline. (2013, Augustus 26). *Het vissengraatdiagram uitgelegd*. Opgehaald van Sigmaonline: https://www.sigmaonline.nl/2013/08/visgraatdiagram/

Stackify. (2019, Maart 26). *What is SDLC*. Opgehaald van Stackify: https://stackify.com/what-is-sdlc/

Stichting Nederlands Normalisatie-Instituut . (1994, maart 13). *NEN-ISO 8402:1994.* Opgehaald van Stichting Nederlands Normalisatie-Instituut : https://www.nen.nl/NEN-Shop/Norm/NENISO-84021994-nl.htm

The open group. (2019, Februari 22). *Archi – Open Source ArchiMate Modelling*. Opgehaald van Archi: https://www.archimatetool.com/

Thinkwise B.V. (2017). *Projectmethodiek v1.12.* Apeldoorn.

Thinkwise Software B.V. (2019, Februari 18). *About thinkwise*. Opgehaald van Thinkwise Software: https://www.thinkwisesoftware.com/en/about-thinkwise/

Trienekens, J. (1994). *Tijd voor kwaliteit.* Amsterdam: Thesis publishers.

Trienekens, J. J. (1994). *Tijd voor Kwaliteit.* Amsterdam: Thesis Publishers .

van Dam, K. (2019, Februari 25). *GAP-analyse*. Opgehaald van Markteffect: https://www.markteffect.nl/meer/kennisbank/GAP-analyse

van der Hoek, J. (2018, Mei 16). *Pursuing a Full Agile Software Lifecycle*. Opgehaald van Mendix: https://www.mendix.com/blog/pursuing-a-full-agile-software-lifecycle/

van der Pols, R. (2009). *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: Van Haren Publishing.

van der Pols, R. (2009). Continuïteitsbeheer. In R. van der Pols, *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: Van Haren publishing.

van der Pols, R. (2009). Impactanalyse. In R. van der Pols, *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: Van Haren Publishing.

van der Pols, R. (2009). Implementeren. In R. van der Pols, *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: Van Haren Publishing.

van der Pols, R. (2009). Kwaliteitsmanagement. In R. van der Pols, *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: Van Haren Publishing.

van der Pols, R. (2009). Ontwerp. In R. van der Pols, *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: van Haren Publishers.

van der Pols, R. (2009). Programmabeheer en distributie. In R. van der Pols, *ASL 2 - Een framework voor applicatiebeheer.* Zaltbommel: Van Haren publishers.

van der Pols, R. (2009). Realisatie. In R. van der Pols, *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: van Haren publishers.

van der Pols, R. (2009). Testen. In R. van der Pols, *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: Van Haren Publishers.

van der Pols, R. (2009). Wijzigingenbeheer. In R. van der Pols, *ASL 2 - Een framework voor applicatiemanagement.* Zaltbommel: Van Haren Publishers.

van Vliet, V. (2019, maart 15). *Total Quality Management (TQM)*. Opgehaald van Toolshero: https://www.toolshero.nl/kwaliteitsmanagement/total-quality-management/

Wong Tiky, Y. (2016). *Software Development Life Cycle.* Hong Kong: The Hong Kong University of Science and Technology.

Zhang, Z. (1997). *Developing a TQM Quality Method Model.* Groningen: Universiteit van Groningen.

Zijlstra, W. (2010, december 20). *Een kwaliteitsmanagementsysteem is geen papierwinkel*. Opgehaald van ZBC Kennisbank: https://zbc.nu/management/kwaliteitsmanagement-systemen-en-iso-9001/een-kwaliteitsmanagementsysteem-is-geen-papierwinkel/

LIJST VAN FIGUREN

LIJST VAN TABELLEN

LIJST VAN AFKORTINGEN

Afkorting	Betekenis
EFQM	European Foundation for Quality Management
ISO	International Organisation for Standardization
SF	Software Factory
TQM	Total Quality Management
INK	Instituut Nederlandse Kwaliteit
SDLC	Software Development Life Cycle
ERD	Entity Relationship Diagrams
SAMM	Software Assurance Maturity Model
CMM	Capability Maturity Model
ASL	Application Services Library
ITIL	Information Technology Infrastructure Library
BiSL	Business Information Service Library
ASL	Application Services Library
KPI	Kritieke Prestatie-Indicator
GUI	Graphical User Interface

BIJLAGEN

1. Interview Jasper Kloost **Datum:** 5 april

In hoeverre ben jij op de hoogte van het onderzoek die ik doe binnen Thinkwise?
Jasper: Ik weet dat het idee er ligt om een kwaliteitsdashboard te maken die wordt geïntegreerd in de Software Factory. Het dashboard moet het mogelijk maken om in 1 oogopslag inzichtelijk te krijgen, voor zover dat kan, wat de kwaliteit is van een project of een projectversie zonder dat je daar allemaal verschillende schermen voor door hoeft te gaan. Een lifecycle is natuurlijk wel veel breder dan dat, dan heb je het natuurlijk ook over beheer, productie en change-management of issues-verzamelen. Dus het is goed om te onderzoeken welke zaken je hier in meeneemt.
Wordt er binnen jouw afdeling op het moment nog gewerkt aan deze metingen of kwaliteitszaken?
Binnen Product Innovation zijn we bezig met het updaten van het Requirements & Design gedeelte. Op het moment is het heel erg op de theorie gestoeld; dus hoe breng je Requirements in kaart en hoe kom je met die requirements via dat design dan tot een Software Factory model. We willen dat het liefste van elkaar los trekken. Je hebt dan echt een requirement-management gedeelte en een stuk work-management. Denk hierbij aan Sprints en Kanbanboards en dat soort dingen. Hierbij kan je het ontwikkelproces zelf meer structureren. Dit is dan ook de plek waar je testbevindingen in zet of bugs-registreren.
Dus zo kan je ook met de software registreren of je requirement voldoet?
Bijvoorbeeld ja. Ik weet niet in hoeverre dat voor jouw project belangrijk is. Maar op deze manier gaat het wel mogelijk zijn om requirements met de software meetbaar te maken en mee te nemen in je dashboard.
Met Mark van den Berg heb ik al een aantal metingen besproken die hij graag in het dashboard zou willen zien. Bijvoorbeeld; het bijhouden wat het percentage van uitgevoerde validaties. Zou dit volgens jou ook een goede toevoeging zijn?
Ja, als je kijkt naar het dashboard is het inderdaad goed om een overzicht te creëren van alle kwaliteits-gerelateerde metrieken die zijn te meten door de Software Factory. Dan kan je bijvoorbeeld denken aan validaties, dat is natuurlijk een hele triviale, dus; staan alle validaties op groen?, je kan bijvoorbeeld kijken of alle control-procedures zijn gereviewd, de verhouding tussen code en commentaar, of er voor alle codes testcases zijn. En misschien wil je ook wel meten of alle features zijn gebouwd in het project. Is er bijvoorbeeld een gebruiker die nergens een eigen schermtype, of formuliergroepen gebruikt.. Dat is misschien ook iets wat je wilt registreren omdat anders mensen niet het onderste uit de kan halen qua mogelijkheden van de Software Factory.
Dus dan zou je ergens moeten aangeven bij welk proces, welke functionaliteiten moeten worden gebruikt?
Je zou inderdaad dan het proces kunnen splitsen tussen datamodelleren, GUI-modelleren. Ik denk dat dat goed zou zijn om te doen. Dus; per fase of per type model een aantal KPI's definiëren die je terug laat komen in het dashboard.
Als ik jou zou vragen: Wat is voor jou Kwaliteit?
Voor mij is kwaliteit, en zeker in de software development; als een product werkt zoals het ontworpen is, en voldoet aan de requirements. Dan heb je hoge kwaliteit. In principe heb je dan ook weinig fouten, want als je fouten hebt is het per definitie niet volgens de requirements. Maar als je iets ontwikkeld wat op zichzelf foutloos is maar niet doet wat de klant/gebruiker gevraagd heeft heb je ook geen kwaliteit.
Dus de gebruiker bepaald de mate van kwaliteit, of de persoon die de requirements opstelt..
Dat is wel de basis inderdaad. Als de requirements al verkeerd zijn kun je nooit hoge kwaliteit

opleveren. De requirements moeten daarbij een goede afspiegeling zijn van wat de gebruiker wil en de software moeten de requirements op een goede manier implementeren; zonder fouten, acceptabele performance, goede user experience etc.

2. Interview Harold Soepenberg

Allereerst: Wat is je naam, functie en afdeling? **Datum:** 15 April
Harold Soepenberg, Thinkwise Project Manager bij Customer Solutions

Aan welke projecten heb je de afgelopen tijd gewerkt binnen Thinkwise?
Vanaf vorig jaar: BOAL, Van Dorp, VDL, REA en nu weer VDL

Dat zijn aardig wat projecten in een korte tijd!
Ja! Ik werk liever aan de wat kortere projecten in plaats van de grote!

Als we praten over kwaliteit praten we vaak over een subjectief begrip. Wat is volgens jou kwaliteit? Als je het plaatst in het perspectief van Softwareontwikkeling
Het is inderdaad subjectief, het is veelomvattend. Maar kwaliteit is naar mijn mening; Goede medewerkers, houden aan afspraken. Software goed doortesten. Foutloze software bestaat niet. Bepaal dan ook het risico van de functionaliteit. Bij sommige functionaliteiten is het risico nihil, maar bij sommige worden er bijvoorbeeld ook records weggegooid. Dit zijn kritieke functionaliteiten want als je hier een fout maakt ben je gewoon je data kwijt. Een goede risicoschatting is hier belangrijk.

Maar als je kijkt naar software, wanneer heeft dat kwaliteit?
Dat is lastig te stellen. Kwaliteit is daarin ook een beleving. Een klant kan een stuk software goed vinden, en het kan weinig fouten bevatten (want als er veel fouten in zitten vindt de klant dit al niet goed). Ik wil alles getest hebben. Weinig aannames. Begrijpen de makers de software.

Heeft kwaliteit dan te maken met Requirements? Dus; in hoeverre voldoet de software aan de requirements?
Ik denk dat kwaliteit niet direct te maken hebben met de requirements. De requirements zorgen ervoor dat jij het juiste gaat bouwen. De klant heeft een beeld van hoe het er uit zal komen te zien. De kans is groter om dit succesvol te doen, als je dit met requirements vastlegt. Als je dit niet doet krijg je achteraf gezeur. Echter, door kwaliteit te bieden heb je de requirements zeker nodig. Het is eigenlijk een must om kwaliteit te bieden, want als je geen requirements hebt, wat ga je dan testen? Wat ga je maken? Dan doe je een best-efford. Maar dan is het testen ook een beetje een best-efford. Dan heb je echter geen garantie of je iets maakt wat de klant daadwerkelijk wil.

Wordt er in jouw projecten een bepaalde methodiek of methode gehanteerd?
We houden ons zoveel mogelijk aan de projectmethodiek van Thinkwise. We werken niet volledig volgens een methodiek. Wél gebruiken we een KANBAN-board. Dus met kaartjes geven we aan wat er moet gebeuren. We zorgen dat het kaartje duidelijk is, en anders werken we deze uit. Vervolgens plaatsen we deze in de verschillende Lanes (To Do, Doing, Done, Ready for Review, Completed).

Maar er is dus geen duidelijk aanwezige methode binnen het project? Alleen verschillende onderdelen en facetten?
Nee, al zou ik dat wel graag willen. Scrum, zoals scrum bedoeld is, past niet volledig in de werkwijze van Thinkwise. Alleen al het aantal projectleden is al niet strokend met hoe scrum moet werken. Wij werken bijvoorbeeld met 2 man. Daarom is het KANBAN-board al voldoende. Als je het hebt over kwaliteit, vind ik het ook belangrijk dat er aan de coding-standards wordt gehouden.

Het waarborgen van die Coding-standards, heb je dit opgenomen in het ontwikkelproces die je hanteert binnen je project?
Nee, als ik code review en ik zie dat er niet aan de standaard wordt gehouden spreek ik diegene daar op aan. Als je het er dan niet mee eens bent, kan je met Frank Wijnhout overleggen of de coding-standards aangepast kunnen worden. Zo niet, dan houdt je je hier aan.

Is er binnen de projecten een gestroomlijnd proces die je doorloopt als je aan het ontwikkelen bent?
Het is niet gestroomlijnd. Als iemand klaar is, gaat die van To-do naar Ready for Review. Dan moet er getest worden en code review gedaan worden. Dan wordt die pas op done gezet. Dit gaat niet altijd op. Je gaat niet continu code review doen want dan blijf je bezig. Als het ene kaartje klaar is kan de volgende alweer gereviewd worden, waar op dat moment soms wel precies dezelfde code in zit. Wat ik daar vaak doe is nagaan of hoelang de code nog niet is gereviewd. Is dit al meer dan 3 weken geleden, dan kijk ik er even naar en dan corrigeer ik waar nodig. Daar is echter geen standaard voor.

Daarnaast heb je wel eens dat je alleen op een project zit. Daar is geen mogelijkheid voor het reviewen van andermans code. In dat geval review ik mijn eigen code nogmaals. Heb ik voldoende commentaar erin staan, als iemand anders ermee aan de slag gaat, begrijpt die het dan.

Komt het vaak voor dat er mensen alleen op een project zitten?
Dat komt regelmatig voor. Als er een project van 4 weken is, en er is verder niemand beschikbaar, dan gaat 1 iemand dat uitvoeren.

Is er dan niet een 2e paar ogen nodig om dit te controleren?
Dat is niet altijd mogelijk. Een buitenstaander kan niet altijd een ander zijn code reviewen. Je weet dan niet wat de code doet, je kent het proces niet, je kent de applicatie niet. En dan moet je iemand anders zijn code gaan reviewen? Het enige wat je kan doen is kijken of de code voldoet aan de coding-standards, is het goed uitgelijnd, staat de join wel goed, begrijp ik wat er gebeurt?, Doe ik iets raars in mijn code en is dat dan opgelost en uitgelegd met commentaar? Dus; is de code goed, performed hij goed en begrijp je het over 3 maanden ook nog?

Is dit aspect belangrijk voor de kwaliteit van het eindproduct?
Ik denk niet dat dit het belangrijkste is. Het belangrijkste is dat de software doet wat het moet doen. Performance is dan iets minder belangrijk. Je kan je er heel erg op richten en optimaliseren, maar een halve seconde voelt de gebruiker niet. Performance komt later pas naar boven. Als de database groeit. Dan moet de performance pas getuned worden. Er zijn duizend manieren om problemen op te lossen met software, maar onder aan de streep staat dat de software functioneel moet werken, het moet op het goede moment performen en dan mag het van mij part een black box wezen.

Vind de klant performance belangrijker dat het functioneel goed werkt of dat het qua lay-out er goed uitziet?
Dan heb je het weer over klantbeleving. Als je het hebt over de kwaliteit van de ervaring. Dat is minstens net zo belangrijk.

Als je praat over kwaliteit. Heb je het dan over klantbeleving of over technische kwaliteit?
Beide. Als ik een huis koop is een fundering 1 van de belangrijkste zaken in een huis. Die zie ik niet. Dat is een van de belangrijkste zaken onder water waar de klant geen weet van heeft. Dan heb je het over goede code, duidelijk wat er gebeurt als iemand anders het op moet pakken en performance. Je merkt dat na 2 jaar je huis gaat schuiven of de muur breekt af als deze zaken niet op orde zijn. De technische kwaliteit is dus; dat de fundering goed moet staan om de kwaliteit richting de klant te bieden en goede klantbeleving mogelijk te maken. Een goede fundering zorgt dat de applicatie werkt,

dan ervaart de klant eigenlijk niks. Want de software werkt prima. Pas als de fundering niet goed staat, dan ervaart de klant de software in negatieve zin.

Het stuk code-review. Is dit altijd afhankelijk van mensenwerk? Of is dit te automatiseren?
Ik denk dat deels te automatiseren. Sowieso de coding-standards. In principe is de functionaliteit op het moment nog lastig om te automatiseren omdat het vaak berust op keuzes van de ontwikkelaar.

In hoeverre zou het dan mogelijk zijn om standaarden te bepalen en de code hier doorheen te halen om zo de kwaliteit van de code te meten?
In principe zou dat wel kunnen. Op het moment heb je al die validaties die in de SF zitten. Dan zou je deze dus moeten uitbreiden dat deze ook kijken naar de code. Al zit je dan nog steeds bij het punt dat je alle validaties moet wegwerken, wat nu ook het geval is. Al valt hier natuurlijk wel wat te halen. Je kunt bijvoorbeeld al een melding geven en deze is niet op de PK of FK gezet. Soms is dit bewust, maar de melding wordt al gegeven.

Hoe ervaar jij het overzicht als je een aantal medewerkers onder je hebt? Nu is dat nog een klein project, maar wat gebeurt er als je 15 man onder je hebt werken?
Het overzicht pakken wordt steeds moeilijker. De vraag is wie gaat er reviewen, gaan de mensen elkaars code reviewen? Dat heeft ook weer zijn voor en nadelen.
Ik kan me voorstellen dat het overzicht pakken lastig is als je met 10 man tegelijkertijd aan verschillende modules werkt. Je wil vervolgens dat iemand verantwoordelijk wordt voor zijn eigen module. Ik denk dat als we SCRUM gaan werken, dat het overzicht makkelijk te bewaken is. Daar zijn voor SCRUM allemaal methoden voor.

En hier zit het overzicht vooral in jouw hoofd?
Hier wordt dan wel een KANBAN-board gebruikt. Het is ook een beetje hoe de klant het wil. Is dat nog SCRUMMER dan SCRUM.. Vaak weten klanten zelf niet wat ze willen en willen ze van ons een best-efford. Dan is SCRUM fantastisch. De backlog is precies een overzicht wat er moet komen. Dus overzicht is er in je backlog. Dan kan je het prioriteren en beoordelen, zodat er een planning wordt gecreëerd. Ook kan je met SCRUM perfect je felocity bepalen. Dus hoeveel functionaliteit kan ik in een bepaalde sprint behalen en hoe performen mijn medewerkers.

Waarom wordt SCRUM dan niet toegepast in de ontwikkeltrajecten van Thinkwise?
Nou, SCRUM werkt niet voor alle klanten.

In die kleine projecten; Bepaald de klant, samen met de standaarden van Thinkwise, de mate van kwaliteit in de software?
Ook, maar ook vanuit mijn eigen inzichten. Ik wil graag kwaliteit opleveren. Ik doe bepaalde dingen waarvan ik vind dat die moeten gebeuren voordat ik het oplever.

En dat is nog bovenop wat de klant wil?
Maar dan is het de vraag; Wat wil de klant? Soms geeft een klant aan dat hij een stuk functionaliteit wil. Dan wil het nog wel eens voorkomen dat ik aangeef dat hij dat kan krijgen maar dat wij er wat bijbouwen, want als het fout gaat dan heb ik een log-in systeem nodig. Als ik dat niet doe ben ik daarna twee keer zo veel tijd kwijt om te zoeken waar het probleem zit.

Geef je dit dan ook aan in het gesprek vooraf?
Ja. Zeker.

Na uitleg van een software lifecycle. Heb je het gevoel dat dit aanwezig is in de projecten binnen Thinkwise?
Nee. Naar mijn mening is er een project, die ronden we af. Die wordt bij de klant neergezet. Daarop

volgt een nazorgfase met wat issues, en dan wordt het bij Service & Care neergelegd. Service & Care pakt daarna de kleinere issues of eventueel grotere functionaliteiten op zich. Dat is ook een keuze. De kennis die bij S&C ligt is nihil. Wij hebben de software gebouwd, kennen de processen, kennen de functionaliteiten en keuzes en weten de valkuilen. Dat weet S&C niet. Er is natuurlijk altijd een mogelijkheid om terug te grijpen naar iemand die in het projectteam zat. En in hoeverre dat de kwaliteit ten goede komt betwijfel ik.

Maar als je kijkt naar de software lifecycle. Jullie starten daar de lifecycle toch?
Ja bij Customer Solutions starten we met de requirements ophalen. Overleggen wat de klant daadwerkelijk wil. Dan komt er een schatting uit, wat het gaat kosten. Dan komt de ontwerpfase, maar bij ons zit dat zo kort op elkaar dat het snel richting ontwikkelen gaat. Dit gebeurt vaak dus in de vorm van kleine proof of conceptjes. Als dit is goedgekeurd, gaan we verder. Dit gaat door tot de uiteindelijke oplevering is geweest en de nazorg. Dan knippen we de connectie en dragen we het over naar S&C. Dit heeft voor- en nadelen. De mensen binnen Customer Solutions komen weer vrij en kunnen ingezet worden voor nieuwe projecten, Alle vragen en issues die binnen komen die komen niet meer in het team. Alleen bij S&C zit minder kennis. Kennis van de applicatie of de omgeving. En dat is dan weer een nadeel.

Zijn de processen een gestroomlijnd of is het iets wat kris-kras door elkaar heen gaat omdat je per module oplevert?
Het is sowieso geen gestroomlijnd proces. Ik denk dat iedereen zijn eigen methodiek erop nahoudt; gebaseerd op de Thinkwise-methodiek. Dat is misschien niet verkeerd, maar het is zeker niet gestroomlijnd.

Dat maakt het wel moeilijk te managen lijkt mij..
Dat is zeker moeilijk te managen.

Als je je puur focust op het ontwikkelen van 1 module. Gebruiken jullie dan een watervalmethode?
Het is maar hoe je het zelf wilt gaan bouwen. Dat kan je zelf bepalen.

Dus elke medewerker krijgt de verantwoordelijkheid om zijn eigen proces in te richten?
Ook dit is afhankelijk van de module. De ene module is gemakkelijker te ontwikkelen. Sommige modules zijn lastig omdat ze nog geen idee hebben hoe dit eruit komt te zien. Hier komt dan veel meer overleg bij kijken, en daar zit dan ook meer tijd in. Ik vind het zelf dan ook fijn om verder te gaan met zaken waarvan ik zeker weet dat ze moeten gebeuren. Pas als ik aannames ga doen, dan stop ik.

Wat doet dit gegeven met jouw overzicht als projectleider? Kun jij hier controle op behouden?
Dat denk ik wel. Soms zijn er modules waar de motor hapert. In die gevallen moet je met de klant overleggen en afspraken maken. Wij willen uren maken, we hebben een deadline en de informatie komt niet. Dit kun je kenbaar maken zodat je gaat bepalen hoe je die mensen dan gaat inzetten. Je kan dan zeggen dat bij 2 modules die wel lekker lopen, je daar 2 mannetjes bij zet. Dat ze er even met zijn 3en aan gaan werken zodat dat af is. Soms is het ook lastig om te bepalen wat wel of niet duidelijk is. En als alles duidelijk is, is er dan voldoende werk om iedereen aan de slag te houden, en het is duidelijk, dan heeft het geen nut om een 3^{e} erbij te zetten.

Hoe creëer je dat inzicht?
Het gaat om communicatie. Openheid is daarin ook een essentieel onderdeel van kwaliteit. Wat vind de klant van ons? Wat vind die van Thinkwise? Van de personen in het team? Of van de communicatie? Je kan als bedrijf 6 maanden op een hok zitten en zeggen; Hier heb je het. Of je

probeert die klant zo veel mogelijk mee te nemen. Het voordeel daarvan is dat je open bent naar de klant, het nadeel is dat ook alle bugs en issues inzichtelijk worden. Een ander voordeel is daarbij is dat de je nooit iets maakt waarvan de klant niet weet dat het gemaakt wordt.

Is dat een stuk requirements up to date houden en opstellen?
Daar praat je over, maar het gaat ook over het terugleggen van verantwoordelijkheid van werk. Bij elke keuze wordt de klant betrokken bij die management.

Dus eigenlijk zeg je dat kwaliteit ook een stuk verwachtingsmanagement bevat.
Het is inderdaad de beleving van de klant. Dus niet alleen de kwaltiet van de software maar ook de kwaliteit van de service is daarin heel belangrijk.

Maar de kwaliteit van de service doet weer wat met de kwaliteit van de beleving lijkt mij.
En dat kan met elke klant verschillen. Voor de een is de beleving anders dan voor de ander. De communicatie naar de klant kan ook per klant verschillen. De klant wil het altijd goedkoper. Die heeft liever dat je het in 10 uur minder maakt, dan 10 uur meer.

Kun je mij meenemen in het ontwikkelproces? Wat zijn hierin de fasen?
Requirements analyse (wat wil de klant, en hoe), Daarna worden ontwerpen gemaakt (modelleren), hierna gaan we ontwikkelen (maken, schermopmaak, functionaliteit), testen. Dit deel is een stuk wat steeds weer terug komt. Je zult zien dat je elke keer weer deze cyclus in gaat. Tot het moment dat de hele applicatie staat. Dan kom je in de acceptatiefase. Dan maken we de acceptatiedocumentatie. Hoe ziet de acceptatietest eruit, waar testen we op? Wat zijn de acceptatiecriteria? Terwijl ik eigenlijk vind dat de klant deze criteria moet opstellen. Wat wil jij nou accepteren? Want als wij dit maken nemen wij rekening met de manier waarop wij de applicatie hebben gerealiseerd. En vanuit mijn ervaring weet je dat als de klant die gemaakte acceptatietest gaat uitvoeren hebben wij dan van te voren die test al gedaan. Die gaat sowieso goed.

Valt de acceptatietest nog binnen Customer Solutions?
Dat valt zeker nog binnen onze afdeling. Wij leveren het werk vanaf de requirements tot en met het werkelijk in productie draait. Dan volgt er nog een stuk nazorg. Als die nazorg nihil wordt, dragen we het over naar Service & Care. Dan zorgen we dat alle gegevens die bekend zijn, in een document worden gezet.
Over welke gegevens heb je het dan?
Servernamen, IP-adressen, procesbeschrijving, risico's, uitzonderingen, klantinformatie, contactgegevens.

Mis je nog een stuk binnen een project?
We missen nog wat op het gebied van testen. Het is nu testen on the fly. Dus je maakt allemaal kleine units (stukken functionaliteit die je net gemaakt hebt) en die laat je doortesten door iemand anders of je test het zelf nog even goed door.

Testcases kunnen gedaan worden binnen de SF om de software automatisch te testen. Gebruik je dit?
Nee dit wordt niet gebruikt. Op het moment testen we handmatig. Het is niet volledig, omslachtig. En het is een stuk onwetendheid. Mensen weten niet hoe het werkt of dat het er überhaupt is. Daarnaast moet je bij elke wijziging de testcases aanpassen, en daar zit gewoon heel veel tijd.
Ook moet je de afweging maken of de klant wel 30% meer tijd wilt besteden aan het testen van de software.

In de SF zit ook Automatische code-review een automatische code-review. Wordt dit gebruikt?
Ik weet nergens vanaf. Dat is voor mij nieuw. Ik moet dat eerst even opzoeken. Dit is natuurlijk een stuk van aandacht. Het is onwetendheid. Waarom worden deze functionaliteiten niet gebruikt?l

Maar wat zou het gebeuren als je deze controles automatisch in je proces meeneemt?
Dan gaat de klant automatisch meer betalen omdat er heel veel tijd in zit. Het is daarom belangrijk dat je van te voren het risico bepaald. Als je per procedure kan aangeven wat het risico is, hoef je deze bijvoorbeeld niet mee te nemen in je testing. Dat scheelt dan weer tijd. Dit verschilt ook per klant. Bij sommige klanten is het belangrijk om alles door te testen en is geld daarin geen issue.

Dus eigenlijk wat je stelt is dat voor de meeste klanten het voordeliger is om issues op te lossen dan van te voren goed door te testen?
Nee dat zeg ik niet. Daarin moet er afgewogen worden wat het risico van de issue is. Als er data wordt weggegooid dan is het altijd een groot risico en die issues moeten het beste worden getest. Het is alleen de vraag hoe je dit doortest. Ik zit daarin te wachten op het plan van Benjamin. En die resultaten kan jij weer goed gebruiken. Wat je wel kan doen, is de validaties in de SF, een kwaliteitsmeting doen.

Zou dat behulpzaam zijn?
Als dat in een dashboard zit wel. Zelf kijk ik daarvoor in de SF. Je moet wel onderscheid kunnen maken wat essentieel is. Ik maak mij niet zo druk om controleprocedures die niet goed zijn gekeurd als je aan het ontwikkelen bent. Want ik keur hem goed, dan moet ik een opmerking geven, wat heb ik gedaan? Is ie goedgekeurd? En een uur later is er een aanpassing gemaakt en staat die weer open.

En bijvoorbeeld; Wat is het percentage van codes die niet zijn gereviewd? Wat zegt dat bijvoorbeeld over de kwaliteit van je product?
Naar mijn mening moet je dan kijken naar de datum van de laatste wijziging. Als iets al zes maanden geleden is aangepast maar nog niet is gereviewd, dan denk ik dat je een punt hebt en deze gereviewd moet worden. Er is echter niet altijd tijd om te reviewen of een procedure bestaat uit meerdere blokken en is dan nog niet volledig af. Control-procedures die 2 weken open staan, wordt waarschijnlijk nog aan gewerkt. Er valt zeker wat uit te halen. Codes die lang niet zijn gewijzigd en die nog niet gereviewd is, daar moet een melding van worden gemaakt.

3. Interview Gert Groenenveld

Allereerst je naam, functie en afdeling!
Gert Groeneveld, Sr. Thinkwise Specialist, Customer Solutions

Zijn er op het moment projecten die je leid?
Ja, we zijn bezig met SAAN, De Jong Verpakking en ik begeleid een software producent.

Ik ben bezig met het verbeteren van de kwaliteit. Dit is vaak een subjectief begrip. Wat is voor jou kwaliteit? Plaats dit maar in de context van Software Development
Kwaliteit van software is voor mij dat de software zo goed mogelijk is afgesteld op de requirements en zo min mogelijk fouten bevat.

Als je kijkt naar jouw ervaring binnen Thinkwise. Wat heeft in het verleden ervoor gezorgd dat je praatte over kwalitatief hoogstaande software?
Dat weet ik eigenlijk niet zo goed. Dat is een gekke vraag aangezien ik voor Sligro heb gewerkt en eigenlijk in 5 jaar tijd geen succesvolle software heb opgeleverd.
Wat ik wel tegenkwam bij De Jong. Blijkbaar snap ik goed van wat zij willen. In gesprek met ze en

dat vertalen naar de software. Dus de juiste User interface, het juiste model daarachter. Dat is ook een stuk wat ik wel kwaliteit vindt; niet zo zeer op de SF maar wel dat je klant begrijpt, daar iets op bouwt en dat ze vervolgens verrast worden door de pro-activiteit van de softwarebouwer. Proberen mee te denken met de klant. Dit beïnvloed dan ook de kwaliteit.

Het gaat dus om de vertaalslag van de wensen naar tastbare software.
Ja! Een deel, maar het moet er onder de motorkap ook goed uitzien. Als jij een keuken besteld, en de keuken ziet er mooi uit maar je trekt een la open en er zit geen bodem in, en je zet de vaatwasser aan maar er komt allemaal water uit. Ook dit gebeurt in de SF ook.

We hadden bij De Jong verpakking bijvoorbeeld een probleem. Ik moest iets doen met de tabel Emballage. Dus ik dacht dat het iets te maken had met pallets of met dozen. Dat was het niet, het was juist hoeveel pallets er binnen kwamen. Dus er kwam iets binnen met emballage daar achter. Dat bleek dus niet om de pallets zelf te gaan, maar om de hoeveelheid pallets. Specificatie wat er werkelijk binnen gekomen was. Niemand heeft ooit de moeite genomen om die tabel te hernoemen. Ik, als nieuw project lid, heb geen idee waar die tabel over gaat. Want ik wordt volledig op het verkeerde spoor gezet.

Dus je praat ook over de kwaliteit van werkwijze?
ja. Het kwaliteit van werkleven. Dat doe je met het hele team. Je hebt de wensen van de klant waar je aan moet voldoen en de kwaliteit van software onder de motorkap, daarnaast heb je nog de wijze waarin het over te dragen is naar Service & Care. En dat zij snappen hoe de software werkt.

Gebruik je binnen je projecten een bepaalde methodiek of managementmethode?
We gebruiken de Thinkwise-methodiek. Bij Sligro hadden ze een eigen methodiek. Dit begon met waterval, scrum en uiteindelijk heel agile. Tegenwoordig is elk project anders en rol ik overal een beetje in.

In hoeverre bepaal je overzicht tijdens je project? Hoe pak je dit overzicht?
Dat is soms lastig. Als je wat grotere projecten leid, dan ben je als projectleider veel tijd kwijt aan het doen van Code Review. Het gegeven om dan te kijken hoe het met het project staat vergt dan ook veel tijd.

Kan je daarin een handvat gebruiken?
Er zit in de SF al wel wat mogelijkheden om het overzicht te pakken en het een en ander aan informatie op te halen. Als jij een template zal maken en je zet hem op gereed, dan moet ik hem nog goedkeuren.

Daarvoor moet je toch wel naar het scherm?
Ja dat klopt. Daarvoor moet je wel een handeling uitvoeren. Maar als jij een validatie draait, en als je synchroniseert met de IAM dan meldt die het ook.

Zal je daarin het fijn vinden om alle resultaten van de SF naast elkaar te zien.
Jazeker. Op het moment weet je bijvoorbeeld niet of de testscripts gemaakt zijn. Ik zou wel willen weten hoeveel procedures er gereviewd moeten worden. Het datamodel wordt lastiger. Want hoe beoordeel je of het datamodel goed is of niet?

Heb jij voor jezelf al een manier gecreëerd hoe je de status van een project kan achterhalen?
Nee, dat moet ik puur doen met 10 verschillende middelen. Je zou inderdaad een scherm moeten hebben als je de SF opent. Met daarin ook de resultaten van afgelopen dagen.

Het testen is wel een belangrijk punt. Gebruik je de testcases binnen de SF?
Zelden. Het is omslachtig. Je bouwt iets. En als je een wijziging doen op het scherm dan klopt je test niet meer. Dan moet je die test opnieuw maken. Je kan dus niet meerdere testen achter elkaar doen. Terwijl je eigenlijk per unit dat wil doen. Dus ik wil een klant opvoeren, dan een order, dan een factuur draaien en dan de verzending. Dan heb je 4 taken die je achter elkaar moet testen. Dan kan je het hele deel goed testen. Nu moet je alles bij elkaar in 1 script testen. Dan moet je het script opnieuw maken.

Dus als je een wijziging doet, dan moet je gelijk het script automatisch aanpassen.
Ja, maar dat lijkt me erg lastig.

Er zit in de SF ook een automatische SQL controle
Is dat de nieuw? Is dat Syntax highlight? Daar ben ik niet van op de hoogte. Nog niet gezien.

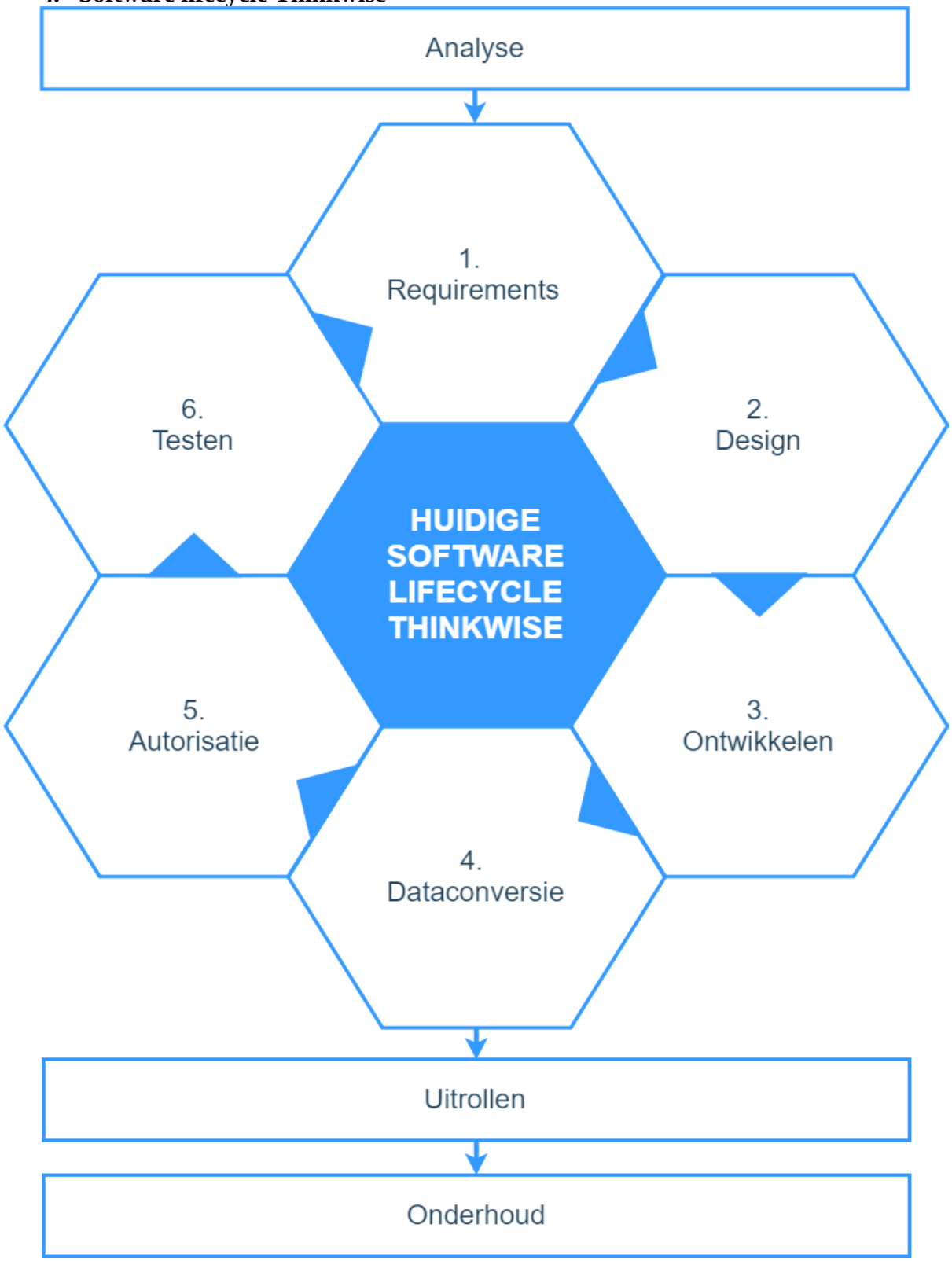
Vaak zijn er ook nog veel projecten die op de oude versie van de SF draaien. Bij Sligro zaten we vorig jaar nog op de G9.6. Het kost zoveel tijd om up te graden. Vaak willen ze daar geen tijd aan besteden en dus ook geld. Dat gebeurd dan ook niet. En bij PI zijn ze ondertussen al 3 stappen verder. Het is voor bedrijven dan ook niet mogelijk om dit te doen. Het kost soms maanden, misschien wel een jaar, om een nieuwe SF te introduceren bij een klant.

Is dat omdat er zoveel data in zit?
Nee het is een grote slag aan werk. Alles moet getest worden, de organisatie moet mee.

Maar om de paar maanden komt er een nieuwe SF uit.
Ja, als je een beetje een kleine omgeving hebt dan wil het wel.

Werken jullie met KANBAN boarden?
Meestal wel, maar bij kleinere projecten overleg je gewoon met elkaar. Het zal mooi zijn om dit in de SF te doen. Het is ook een probleem bij de requirements. Hoe gedetailleerd moet je dit hebben. Je bent in projecten vaak zo lang bezig met het opstellen van de requirements voordat je kan beginnen met ontwikkelen. En dan moet je vaak nog je functionaliteiten beschrijven. Het is allemaal hartstikke handig zijn, om dit beter te doen maar dit kan bezig.

4. Software lifecycle Thinkwise



5. Procesvergelijking Analysefase

TABEL 2: PROCESVERGELIJKING ANALYSEFASE

Proces Thinkwise	Proces Impactanalyse ASL (van der Pols, Impactanalyse, 2009)	Proces Ontwerp ASL (van der Pols, Ontwerp, 2009)
Projectanalyse uitvoeren	Het in kaart brengen en uitwerken van de wijzigingen behorende bij een release (aannemen). Het in kaart brengen van de producten die worden geraakt en de bijbehorende veranderingen na de wijziging.	Diepgaande analyse van de vraagstelling of aangegeven wijzigingen, Vertaling van de informatie naar de gegevensvragen, In kaart brengen van de relevante delen in het systeem.
Valideren bij de klant	Verificatie van de impactanalyse	Terugkoppeling naar opdrachtgever, Akkoord door opdrachtgever
Modules Creëren en volgorde bepalen	Het in kaart brengen van de relaties tussen verschillende voorgestelde wijzigingen en andere releases. Opstellen scenario om wijzigingen aan te brengen	Bepalen mogelijke oplossingsrichtingen, Kiezen van een oplossingsrichting
Schatting project	Inschatten effecten op langere termijn, Begroten van de omvang van de activiteiten als gevolg van de veranderingen en inschatten van de tijdsaspecten.	
Projectplan opstellen	Inschatten benodigde middelen en werkwijzen, Bewaken voortgang, middelen en proces	
Review door collega		

6. Procesvergelijking Requirements-fase

TABEL 3: PROCESVERGELIJKING REQUIREMENTS-FASE

Proces Thinkwise	Proces Ontwerp ASL (van der Pols, Ontwerp, 2009)	Proces Realisatie (van der Pols, Realisatie, 2009)
Controleren of Requirements gehaald zijn	Bewaking van de voortgang, Evalueren van de voortgang	Planning van de realisatie, Bewaken voortgang en maken voortgangsrapportage, Evalueren en bepalen van verbeteringsvoorstellen
Systeem-requirements in kaart brengen	In kaart brengen van relevante delen van het systeem	Ontwerpen Technische oplossing (en alle onderliggende activiteiten), Bepalen technische impact (en alle onderliggende activiteiten)
User-requirements in kaart brengen	Vertaling van vraagstelling naar gegevensbehoefte, Opstellen functionele testspecificaties, Naderende uitwerking vraagstelling (en alle onderliggende activiteiten)	
Valideren bij de klant	Interne kwaliteitsborging, Terugkoppeling naar opdrachtgever, Akkoord door opdrachtgever	

7. Procesvergelijking Designfase

TABEL 4: PROCESVERGELIJKING DESIGNFASE

Proces Thinkwise	Proces Ontwerp ASL (van der Pols, Ontwerp, 2009)
Designspecificaties opstellen	Bepalen van de mogelijke oplossingsrichtingen Kiezen van een oplossingsrichting
Koppelen met requirements	
Mock ups maken	Bepalen en uitwerken specificaties naar een ontwerp

8. Procesvergelijking Ontwikkelfase

TABEL 5: PROCESVERGELIJKING ONTWIKKELFASE

Proces Thinkwise	Proces Realisatie ASL (van der Pols, Realisatie, 2009)
Datamodelleren	Realiseren oplossing (en alle onderliggende activiteiten)
Functionaliteit maken	Realiseren oplossing (en alle onderliggende activiteiten)
Schermopmaak	Realiseren oplossing (en alle onderliggende activiteiten)
Testen van de functionaliteit	Testen oplossing (en alle onderliggende activiteiten)
Review door collega	

9. Procesvergelijking Dataconversie-fase

TABEL 6: PROCESVERGELIJKING DATACONVERSIE-FASE

Proces Thinkwise	Proces Impactanalyse ASL (van der Pols, Impactanalyse, 2009)	Proces Realisatie ASL (van der Pols, Realisatie, 2009)	Proces Programmabeheer en distributie ASL (van der Pols, Wijzigingenbeheer, 2009)
Validaties uitvoeren	In kaart brengen wijziging (en alle onderliggende activiteiten) Inschatten verandering (en alle onderliggende activiteiten) Inschatten consequenties (en alle onderliggende activiteiten)		Onderkennen van mogelijke ‘interferenties’ van de verschillende change sets. Vaststelling en validatie van gerealiseerde uitgangspunten van de release.
Validaties oplossen		Aanpassen programmatuur	
Wijzigingen uitrollen naar database		Aanpassen gegevensobjecten	Bepalen van de change package. Overzetten naar productie (en alle onderliggende activiteiten)
Nieuwe versie aanmaken			Registratie objecten in het onderhoudsproces (en alle onderliggende activiteiten) Bewaren van versies van objecten Het definiëren van change sets

10. Procesvergelijking Testfase

TABEL 7: PROCESVERGELIJKING TESTFASE

Proces Thinkwise	Proces Realisatie ASL (van der Pols, Realisatie, 2009)	Proces Testen ASL (van der Pols, Testen, 2009)
Unittest inrichten		
Unittest uitvoeren	Testen oplossing (en alle onderliggende activiteiten)	
Gebruikerstest uitvoeren		Functionele (logische) systeemtest (en alle onderliggende activiteiten) Technische systeemtest (en alle onderliggende activiteiten)

11. Procesvergelijking Uitrolfase

TABEL 8: PROCESVERGELIJKING UITROLFASE

Proces Thinkwise	Proces Implementeren ASL (van der Pols, Implementeren, 2009)
Bepalen uitrolactiviteiten	Voorbereiden van en ondersteunen bij het inplannen van de gegevensverwerking Voorbereiden van de feitelijke opdrachtverstrekking tot overzetten naar exploitatie
Acceptatietest ontwikkelen	Ondersteuning gebruikersorganisatie (en alle onderliggende activiteiten)
Deploymentplan opstellen	Ondersteunen bij de voorbereiding van de verwerking en installatie Ondersteunen bij of uitvoeren van het veranderen van de gegevensdefinities op de productieomgeving, begeleiden van de technische conversies.
Upgrade draaiboek opstellen	Archiveren van stukken, voorbereiden akkoordmelding, opstarten evaluaties Afronding opdracht (en alle onderliggende activiteiten) Besturing (en alle onderliggende activiteiten)
Laatste controle	Aanpassen van de fouten naar aanleiding van de acceptatietest

12. Automatisering en wijze van de metingen in de SF

TABEL 9: AUTOMATISERING METINGEN IN SF

Fase	Meting	Automatisering mogelijk	Wijze van meten	Beschikbaar in huidige versie SF
Analyse	Bevat de projectanalyse alle onderdelen?		Checklist	
Analyse	Bevat het projectplan alle onderdelen?		Checklist	
Analyse	Is de projectanalyse gevalideerd bij de klant?		Controle	
Analyse	Is de projectanalyse gereviewd door een collega?		Controle	
Requirements	Zijn de Requirements gevalideerd bij de klant?		Controle	
Requirements	Zijn de Requirements gehaald?		Query	
Requirements	Is de testspecificatie volledig beschreven?		Controle	
Design	Zijn de designspecificaties gekoppeld met de requirements?		Query	
Ontwikkelen	Zijn er fouten gevonden in het datamodel?		Query	
Ontwikkelen	Zijn er fouten gevonden in de functionaliteit?		Query	
Ontwikkelen	Zijn alle functionaliteiten (codes) getest?		Query	
Ontwikkelen	Zijn alle functionaliteiten gereviewd door een collega?		Query	
Dataconversie	Zijn alle validaties opgelost?		Query	
Dataconversie	Is er een nieuwe versie aangemaakt na de vorige databasewijziging?		Query	
Autorisatie	Zijn alle rollen volledig ingericht?		Query	
Testen	Zijn alle functionaliteiten gedekt met een unittest?		Query	
Testen	Zijn alle unittests succesvol afgerond?		Query	
Uitrollen	Is de acceptatietest volledig uitgevoerd?		Controle	
Uitrollen	Bevat het deploymentplan alle onderdelen?		Checklist	
Uitrollen	Bevat het Upgrade-draaiboek alle onderdelen?		Checklist	
Uitrollen	Is alles klaar voor productie?		Query	

13. Schatting benodigde uren voor realisatie dashboard

TABEL 10: SCHATTING BENODIGDE UREN REALISATIE DASHBOARD

Fase	Meting	Schatting realisatie (uren)
Analyse	Bevat de projectanalyse alle onderdelen?	8
Analyse	Bevat het projectplan alle onderdelen?	8
Analyse	Is de projectanalyse gevalideerd bij de klant?	2
Analyse	Is de projectanalyse gereviewd door een collega?	2
Requirements	Zijn de Requirements gevalideerd bij de klant?	2
Requirements	Zijn de Requirements gehaald?	4
Requirements	Is de testspecificatie volledig beschreven?	2
Design	Zijn de designspecificaties gekoppeld met de requirements?	2
Ontwikkelen	Zijn er fouten gevonden in het datamodel?	2
Ontwikkelen	Zijn er fouten gevonden in de functionaliteit?	2
Ontwikkelen	Zijn alle functionaliteiten (codes) getest?	2
Ontwikkelen	Zijn alle functionaliteiten gereviewd door een collega?	2
Dataconversie	Zijn alle validaties opgelost?	2
Dataconversie	Is er een nieuwe versie aangemaakt na de vorige databasewijziging?	2
Autorisatie	Zijn alle rollen volledig ingericht?	4
Testen	Zijn alle functionaliteiten gedekt met een unittest?	2
Testen	Zijn alle unittests succesvol afgerond?	2
Uitrollen	Is de acceptatietest volledig uitgevoerd?	8
Uitrollen	Bevat het deploymentplan alle onderdelen?	8
Uitrollen	Bevat het Upgrade-draaiboek alle onderdelen?	8
Uitrollen	Is alles klaar voor productie?	2
	Totaal:	76 uur

14. Stappenplan implementatie van de oplossingen

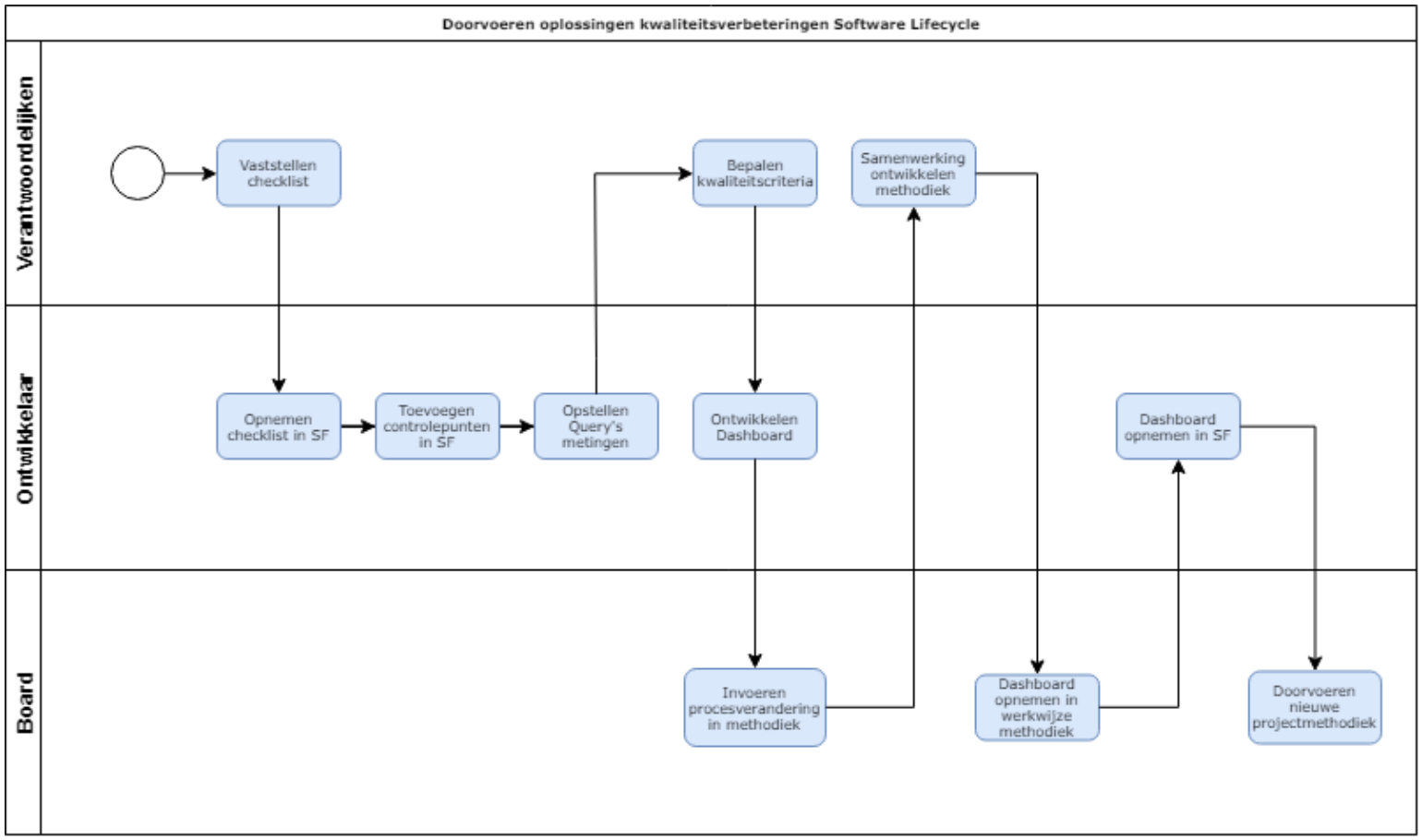
In het volgende document zijn de benodigde stappen in de juiste volgorde beschreven om de opgestelde verbeteringen te implementeren in de organisatie.

1. **Vaststellen van checklists voor meting van volledigheid van documentatie**
   Als het gaat om de volledigheid van de opgestelde documentatie in een project zal dit door middel van een checklist moeten worden geëvalueerd. Dit komt omdat het opstellen van de checklist niet in de SF gebeurt, en om die reden niet automatisch kan worden geëvalueerd. Om de checklist samen te stellen moeten er eerst besloten worden aan welke zaken de documentatie moet voldoen en of dit per klant/afnemer kan verschillen. De checklist moet worden opgesteld voor de volgende documenten: Projectanalyse, Projectplan, Deploymentplan en het Upgrade-draaiboek.
2. **Opnemen checklist voor volledigheid van documentatie in SF**
   Na het vaststellen van de inhoud van de checklist is de volgende stap om deze checklist op te nemen in de SF. Dit kan door een simpele checklist en het afvinken van de bijbehorende componenten. In het geval van een vaste set aan componenten voor elk project kan er door een selecte groep verantwoordelijke een lijst met componenten worden vastgesteld. In het geval van een flexibele set aan componenten, kan er een volledige lijst met componenten worden vastgesteld waar vervolgens per project aangegeven kan worden aan welke componenten de documentatie in dat project moet voldoen. Dit zal betekenen dat ook per project het vaststellen van de documentatiecomponenten bij de analysefase hoort.
3. **Toevoegen missende controlepunten in SF**
   Sommige opgestelde metingen zijn in de vorm van een controle. Dit wil zeggen dat er een simpele check plaats vindt om te controleren of een actie wel of niet is uitgevoerd. Dit is in de SF te integreren door een taak of kolom toe te voegen die registreert of er een actie is uitgevoerd. De check komt als extra kolom in de tabel waar de gegevens van dat proces worden opgeslagen. De checks gaan over de volgende processen: Validatie klant op projectanalyse, review projectanalyse door collega, validatie klant op requirements, volledigheid van testspecificatie, volledigheid van uitvoering acceptatietest.
4. **Opstellen query's voor automatische metingen**
   Nu alle processen in de software lifecycle meetbaar zijn, kunnen de query's worden ontwikkeld om de resultaten van de metingen inzichtelijk te krijgen. Naast de resultaten van de checklists en controles bij bovenstaande stappen met een query in een KPI weer te geven, is het ook nodig om de overige metingen op te halen. De data die nodig zijn voor deze metingen is te vinden op verschillende plekken in de SF. De query's moeten gaan over de metingen in Tabel 9 met het kopje: Query bij de kolom: Wijze van meten.
5. **Bepalen en inrichten kwaliteitscriteria voor opgestelde metingen**
   Als alle functionaliteit is geschreven om de metingen uit te voeren, is de volgende stap om de kwaliteitscriteria te bepalen en in te bouwen in de SF. Dit is een weergave van het geaccepteerde resultaat van de metingen. De selecte groep verantwoordelijken kan vaststellen waar de metingen aan moeten voldoen. Als dit een vaste set met criteria is, kan dit meegenomen worden in de ontwikkeling van het dashboard. Verschillen deze metingen per klant, dan zal er een criteria-tabel moeten komen om de criteria vast te stellen en de KPI's op te meten.
6. **Ontwikkelen dashboard voor alignmentverbeteringen**
   Als de KPI's zijn ingericht is het de bedoeling dat de resultaten overzichtelijk in een dashboard worden verwerkt. Dit kan tevens door middel van de SF. Hierbij is het goed om op hoofdlijnen de resultaten te laten zien met daarbij de mogelijkheid in te zoomen op een

gedetailleerde specificatie van het resultaat. Het dashboard zou de gebruiker een snel overzicht moeten geven van de algemene kwaliteit van de processen in het project.

7. **Invoeren procesveranderingen in projectmethodiek**
   Als het dashboard is ingericht is het zaak om de wijzigingen in de processen door te voeren. De wijzigingen zijn echter zo nihil dat het kan worden opgenomen in de Thinkwise projectmethodiek. Het veranderen van deze proceswijzigingen is dan ook een kleine ingreep en heeft weinig impact op andere zaken in de methodiek van Thinkwise.
8. **Projectleiders betrekken bij wijzigingen in projectmethodiek**
   Voordat de wijzigingen in de methodiek worden doorgevoerd is het goed om de projectleiders en senior-medewerkers te betrekken bij het ontwikkelen van de methodiek. Zij zijn immers verantwoordelijk voor de uitvoering ervan. Om deze eigenaarschap te creëren moeten deze medewerkers worden betrokken bij de wijzigingen in de methodiek door bijvoorbeeld focusgroepen, brainstormsessies of interviews te organiseren om hen mee te laten denken in de door te voeren wijzigingen.
9. **Dashboard onderdeel maken van werkwijze projecten door op te nemen in projectmethodiek**
   Om ervoor te zorgen dat het ontwikkelde dashboard daadwerkelijk wordt gebruikt in de praktijk is het noodzakelijk om het gebruik hiervan op te nemen in de te hanteren werkwijze. Dit kan bereikt worden door het dashboard dus ook op te nemen in de methodiek. Belangrijk is hierbij dat er goed wordt onderbouwd waarom het dashboard van toegevoegde waarde is voor de uitvoering van succesvol kwaliteitsmanagement in een project.
10. **Dashboard opnemen in SF**
    Als er in de projectmethodiek is beschreven hoe en waarom het dashboard in de werkwijze binnen de projecten is opgenomen is de laatste stap dat het dashboard in de SF wordt opgenomen. Het is belangrijk dat het dashboard op een centrale plek wordt gepositioneerd zodat het gemakkelijk te bereiken is en het overzicht op de kwaliteit snel te vinden is.
11. **Doorvoeren nieuwe projectmethodiek**
    Als alle wijzigingen zijn onderbouwd en opgenomen in de projectmethodiek en het dashboard is toegevoegd aan de SF, kan de projectmethodiek over de gehele organisatie worden doorgevoerd.

Zie onderstaand in Figuur 31 een visuele weergave van het stappenplan.



FIGUUR 31: VISUELE WEERGAVE STAPPENPLAN IMPLEMENTATIE

15. Interview Robert van Tongeren

Interview met: Robert van Tongeren **Datum:** 15 April
Functie: Senior Thinkwise Specialist
Afdeling: Customer Solutions
Taken: Het begeleiden van ontwikkelaars als Lead-Developer.
Werkzaam bij Thinkwise: 5 jaar

Samenvatting interview:
Robert werkt al 5 jaar voor Thinkwise en is vanaf februari 2019 werkzaam in het project voor Zeeman. Hij heeft voorheen het EDI traject gemaakt voor VDL. In het verleden ook werkzaam geweest voor Bussen en Topacs waar, wegens een kleine omvang van project, hij vaak alleen werkzaam was en dus alle rollen en verantwoordelijkheden in het project bekleedde. Bij grotere klanten werkt hij aan modules die met de klant in overleg worden samengesteld, dit duurt wel langer.

Kwaliteit
Kwalitatief goede software is Software wat doet wat de klant verwacht in de tijd dat de klant dit verwacht. Het is het voldoen aan de verwachting van de klant, qua tijd en qua functionaliteit. Het in kaart brengen van deze verwachting is bij de requirements essentieel. De klant moet het idee hebben dat er niet meer te halen valt dan datgeen wat je biedt. In het verleden was verlies in kwaliteit vaak een gevolg van veranderende requirements waardoor de software tussentijds werd aangepast, in plaats van dat het opnieuw werd gemaakt.

Projectmanagement
De projectmethodiek laat Robert afhangen van de projectleider. Hij is meer verantwoordelijk voor het aansturen van de ontwikkelaars. Code review is daarbij een belangrijk punt op het gebied van kwaliteit. Dit is dan zijn verantwoordelijkheid en hij bepaald op dat moment of de code voldoet aan de eisen of requirements. Hier staat de projectleider dan weer los van. De kwaliteit is dan zijn perceptie van de verwachting van de klant. Hij test zelf ook zijn code, en een andere collega evalueert zijn code. Naast het reviewen van code en requirements wordt er verder weinig geëvalueerd in het proces. Het overzicht is te pakken door veel met elkaar te overleggen en elkaar te controleren op het gemaakte werk. Dit is echter niet 1 persoon. Thinkwise werkt echter wel met Trello. Daarbij is enig overzicht te vinden, maar ook niet voldoende. Er is geen overzicht hoe ver je bent in het project of wat de kwaliteit is, van datgene wat je hebt opgeleverd. Er is wel een overzicht van de stand van de designspecificaties. Daarnaast zijn we bij Thinkwise nog wel eens bang om de klant een ‘nee’ te verkopen of uitstel te vragen om kwaliteitsredenen. Soms is dit nodig omdat je als ontwikkelaar weet dat je uiteindelijk er problemen van gaat ervaren. Vaak laten we het er dan maar inzitten zodat wij de software toch kunnen leveren voor de deadline.

Software Lifecycle
Tegenwoordig wordt er steeds meer ervaren dat er een cyclusvorm bestaat in de ontwikkelprocessen. Het aanpassen van de requirements en designspecificaties voordat men ontwikkeld. De ontwikkelprocessen op grote lijnen: Processen in kaart brengen, requirements in kaart brengen, designspecificaties schrijven, ontwikkelen, testen, accepteren. Net zolang tot de test geaccepteerd wordt en dan zet je hem door naar de acceptatietest voordat het naar Service & Care gaat. Daar hanteert men een eigen cyclus.

16. Interview Robert-Jan de Nie

Naam: Robert-Jan de Nie **Datum:** 16 April
Functie: Thinkwise Projectleider
Afdeling: Customer Solutions
Werkzaam bij Thinkwise: 10 jaar

Samenvatting Interview:
Robert-Jan werkt al 10 jaar voor Thinkwise. Recentelijke projecten waar hij aan heeft gewerkt zijn Heuvelman en Vacansoleil.

Kwaliteit
Kwaliteit van software is software die doet wat die moet doen en voor de gebruikers goed te begrijpen is. Een ontwikkelaar moet ook gemakkelijk er mee om kunnen gaan. Dus onder de motorkap moet het goed gedocumenteerd en gestructureerd zijn. De software kan goed zijn als het voldoet aan de requirements, maar die requirements moeten ook kwalitatief goed zijn. Slechte requirements komen vaak voor omdat de klant soms ook niet goed weet wat hij wil. Het gaat om voldoen aan de realiteit. Kwaliteit software is dus voldoen aan kwalitatief goede requirements. Toch wordt er gesproken over kwaliteit als de klant blij of tevreden is en dat het probleem is opgelost. Kwaliteit heeft ook met tijd te maken. Hoe meer tijd je hebt, hoe beter je kan ontwikkelen en hoe meer tijd je hebt om te testen.

Projectmanagement
In de afgelopen projecten is er geen kwalitatief goede software opgeleverd. Dit komt omdat de projectleden zo veel verschillende rollen en taken bekleedde dat er dingen van hen gevraagd werd, wat ze eigenlijk helemaal niet konden waardoor je nooit kwaliteit kan realiseren. Eigen oplossingen verzinnen, om problemen heen werken. Happy flow aanhouden. Onder tijdsdruk produceren, want het moest live. Dan gaat men fouten maken. De Thinkwise methodiek is niet duidelijk. Sluit weinig aan bij de realiteit. Robert-Jan wil graag volledig Scrum of Agile gaan werken. Thinkwise en Scrum sluit goed op elkaar aan. Je wil ook meer ‘branching en merging’. Tevens geeft Robert-Jan aan dat de overgang naar Service & Care beter kan. Dit heeft deels te maken met de structuur van je applicatie en de volledigheid van documentatie. Soms vraagt een applicatie om constante verbetering en gaat dit nooit over naar Service & Care. Dan ga je steeds opnieuw de lifecycle in.
Op het gebied van overzicht vindt Robert-Jan dat dit op het moment te bereiken is door een dagelijkste stand-up te doen. Als dit niet gebeurt, is overzicht lastig te vinden. Hiervoor zou eigenlijk in de SF een tool moeten komen die alles met elkaar verbind en moet Thinkwise meer aandacht besteden aan het ontwikkelen van een bewezen en gedegen projectmethodiek. Er wordt op dit moment heel erg veel afgeweken van de methodiek waardoor mensen hun eigen werkwijze aanhouden. Nu wordt er bijvoorbeeld met de hand getest of door derde programmatuur. We hebben zelfs maar 2 testers, en dit ook al sinds een maand. Dat is 1 tester op 100 ontwikkelaars.

Software Lifecycle
In mijn ogen is het een watervalmethodiek waarbij er van te voren specificaties worden opgesteld. Dan wordt er in modules gebouwd en getest. Tijdens die modules ga je de specificaties weer verdiepen en uitwerken. De ontwikkelprocessen zijn hierbij: Procesanalyse, projectplan opstellen, modules ontwikkelen, klant evalueert dat, uiteindelijk is alles klaar, ga je acceptatietraject in en gaat de applicatie live.

17. Focusgroep De Nie, Soepenberg en van Tongeren

Personen: Robert-Jan de Nie, Harold Soepenberg en Robert van Tongeren
Datum: 27 mei
Afdeling: Customer Solutions

Samenvatting Focusgroep
Het doel van de focusgroep is het bepalen van de kritieke processen in een fase van de analysefase en het opstellen van metingen om deze kritieke processen te evalueren. Aan de hand van de opgestelde software lifecycle wordt dit besproken.

Analysefasen
Fase wordt in samenwerking met Business Development (sales) afgewerkt. **De Nie:** Hier worden dus op hoog niveau al Business en User requirements opgesteld en worden de processen in kaart gebracht. Dit heb je nodig om tot een inschatting van de scope te komen. Hier worden tevens de modules gecreëerd. **Van Tongeren:** Dit is nog niet inhoudelijk. Vaak weten ze het al, maar dat komt later in het proces. **De Nie:** Ik denk dat hier een checklist perfect zou zijn. Dus is de analyse volledig? Heb ik al mijn modules in kaart? Dat soort vragen. **Van Tongeren:** Het is altijd ook afhankelijk van de klant. De klant wil een bepaalde werkwijze. Dus niet elke analyse is voor elke klant hetzelfde. **De Nie:** Vaak weet de klant wel veel informatie van die checklist. **Soepenberg:** Het blijft wel mensenwerk, dus als je die checklist hebt opgesteld dan weet je eigenlijk nog steeds niet 100% zeker of het echt goed is. Er moet gewoon een goed document zijn. Het moet volledig zijn. Een checklist om te kijken of alles erin zit is daarbij een perfecte uitkomst. **De Nie:** Ja! Dus wat zijn de kwaliteitscriteria voor een analyse. Het moet ook nog gereviewd worden. Zodat het overdraagbaar is. **Van Tongeren:** Dan alsnog moet het overeenkomen met de realiteit. **De Nie:** Ja, het moet ook gereviewd worden door de klant. **Van Tongeren:** Dus praat je over kritiek; dan moet de analyse worden opgesteld is het gevalideerd door een collega en is het gevalideerd door een klant. **De Nie:** Ja, pas als dat goed is, ga je de modules en de schatting uitvoeren. **Van Tongeren:** Inderdaad. **De Nie:** Wat je daarna ook nog moet doen is een projectplan maken. Aan de hand met alle vergaarde informatie moet je je project gaan inrichten. **Allen:** Denk aan planning, scope, risico's en grijze gebieden.

Vastgesteld proces en kritieke processen:
Projectanalyse uitvoeren → Review collega → Validatie klant →Modules creëren en volgorde bepalen →Schatting project → Projectplan opstellen

Requirement-fase
Van Tongeren: Eigenlijk is je doel van de requirements opstellen je business requirements en je systeem requirements opstellen is het op een zodanige wijze dat je precies weet wat je moet maken. Toch? **Allen:** ja. **Van Tongeren:** Je brengt in principe in kaart wat je moet maken. Dit ga je definiëren in user requirements en systeem requirements en dat valideer je met de klant. Dat laatste is kritiek, want je software moet overeen komen met de wens van de klant. **Soepenberg:** In een latere fase dan ga je die requirements ook aftikken. Dat wil je weten. Maar dat is eigenlijk als je weer terug komt in de lifecycle. **De Nie:** Het probleem heb je alleen dat, als je dit meetbaar maakt, je vaak niet weet hoeveel requirements je in totaal hebt. Het kan namelijk zijn dat er op een later tijdstip er nog requirements bij komen. **Van Tongeren:** Ik kan je wel vertellen dat er vervolgens niet altijd terug wordt gegaan naar de requirements maar de oplossing wordt aangepast. Of dat de requirements bij productie niet meer voldoen aan de realiteit. Dit kan een groot probleem zijn.

Vastgesteld proces en kritische processen:
Controleren of requirements gehaald zijn → Systeem-requirements in kaart brengen → User requirements in kaart brengen → Valideren bij de klant.

Designfase

Van Tongeren: Designspecificaties opstellen is werkelijk het meest nutteloze wat er bestaat. **De Nie:** Designspecificaties opstellen kost echt belachelijk veel tijd. Het opstellen ervan kost bijna net zoveel tijd als het daadwerkelijk ontwikkelen van die functionaliteit. Ga gewoon aan de gang. Geef mensen vertrouwen. **Van Tongeren:** De ervaren ontwikkelaars kauwen het in principe voor tegen de junioren. Die ontwikkelaars hoeven er zelf niet meer over na te denken. Het is beter om henzelf daar over na te laten denken. **Soepenberg:** Ik gebruik geen designspecificaties. Ik bouw enkel de requirements en vraag feedback van de klant. **De Nie:** Ik ga er vaak wel even voor zitten. **Soepenberg:** In principe zijn de Trello kaartjes mijn designspecs. **Van Tongeren:** De designspecificaties opstellen is niet het probleem. Het is het niveau en de diepte waarin we dat nu moeten doen, wat het niet efficiënt maakt. Kost veel te veel tijd. **Van Tongeren:** Ik moet het voor de zeeman gebruiken in de SF. **De Nie:** Ik gebruik het liever niet. **Van Tongeren:** Je kan echter wel de designspecs koppelen met je requirements. Dus als je op zoek bent naar een kritisch proces is dat er 1. **De Nie:** Ja! Elke designspecificatie moet gekoppeld zijn met minimaal 1 requirement. En na het koppelen maak ik soms nog een Mock-up.

Vastgesteld proces en kritische processen:
Designspecificaties opstellen → Koppelen met requirements → Mock-ups maken

Ontwikkelfase

De Nie: Stel je moet iets gaan bouwen, bijvoorbeeld je gaat een requirement bouwen. Dan begin je met je datamodel want je moet zorgen dat je je gegevens ergens kan opslaan. Dan ga ik eerst zorgen dat het werkt. Dus de functionaliteit. Vervolgens ga je dat scherm opmaken en zorgen dat het er netjes uit ziet en test je wat je gemaakt hebt. **Soepenberg:** Het is ook een iteratief proces. Je doorloopt dit proces constant opnieuw als je een wijziging moet doen. **Van Tongeren:** Daarna moet hier dus een controle worden uitgevoerd op de functionaliteit. **De Nie:** Ik vind het minstens net zo belangrijk of die ontwikkelaar ook zijn datamodel wel goed gebouwd heeft. **Van Tongeren:** Die is kritiek. **De Nie:** Maar de koppeling van al deze dingen met elkaar is er niet. Er wordt te weinig gereviewd. Je kon eerder nog aangeven dat je met deze requirement aan de slag kon. **Van Tongeren:** Je kan zeggen dat je met designspecs aan de slag gaat of klaar bent. **De Nie:** Oh?! Dat wist ik niet. **Van Tongeren:** Als ik zou moeten aangeven welke kritiek zijn dan zou ik zeggen: Datamodelleren, Functionaliteit, Testen en Review. **De Nie:** Daar ben ik het mee eens.

Vastgesteld proces en kritische processen:
Datamodelleren → Functionaliteit maken → Schermopmaak → Testen functionaliteit → Review door collega

Dataconversie-fase

Soepenberg: Dit is eigenlijk heel simpel. De validaties zijn hier de basis. Dat is ook belangrijk voor kwaliteit. Daar zit je foutcontrole. **De Nie:** inderdaad. Je voert gewoon de validaties uit, die los je op als het moet. En dan rol je alles uit. **Van Tongeren:** Je hebt ook nog het maken van een nieuwe versie, redelijk belangrijk. Maar opzich niet kritiek. **De Nie:** Het belangrijkste zit dan natuurlijk in het oplossen van de validaties. Eigenlijk moeten alle validaties opgelost zijn om te spreken van een goede software set.

Vastgesteld proces en kritische processen:
Validaties uitvoeren → Validaties oplossen → Wijzigingen naar database → Nieuwe versie aanmaken

Autorisatiefase

Van Tongeren: Dit deel komt straks in de nieuwe versie van de SF. **De Nie:** Ik vind het eigenlijk een belangrijk aspect van het ontwikkelwerk. **Soepenberg:** Het moet niet zo kunnen dat een junior dat kan doen. **Van Tongeren:** Waarom zou dat niet kunnen? **De Nie:** De ontwikkelaar maakt zelf ook de applicatie. **Van Tongeren:** Het inrichten van rollen heb je, en dat is een ander proces dan het inrichten van rechten of gebruikersgroepen. **Van Tongeren:** Dus je hebt Rollen inrichten en autorisatiewerk. Dat kan een klant doen, of een lead-developer. Want dit is essentieel en belangrijk. **De Nie:** Ik zou wel willen weten of het gebeurt is. **Soepenberg:** Of elke rol minimaal 1 keer gekoppeld is.

Vastgesteld proces en kritische processen:
Rollen aanmaken → Rollen inrichten → Autorisatie inrichten

Testen

Van Tongeren: Nog voordat je de gehele applicatie gaat uitrollen ga je de unit test uitvoeren. **Soepenberg:** Dit doe je constant. **Van Tongeren:** De unittest zit in je cyclus. Daarna ga je al het gemaakte werk testen. **De Nie:** Je doet ook nog een gebruikerstest. Dus je hebt de unittest waarmee je de functionaliteit test en je hebt de gebruikerstest daarbij test een persoon of de requirement klopt. **Van Tongeren:** Dat is gewoon het proces doorlopen. **Soepenberg:** Ik doe alle testresultaten in Excel opschrijven. Waar zet jij je testscenario's dan? **De Nie:** Ik heb geen testscenario's want ik ben geen tester. **Van Tongeren:** Maar je kan pas spreken van een goede test als de test is ingericht en is uitgevoerd. **De Nie:** De gebruikerstest is niet kritisch want het blijft mensenwerk.

Vastgesteld proces en kritische processen:
Test inrichten → Unittest uitvoeren → Gebruikerstest uitvoeren

Uitrolfase

Soepenberg: Daar moet een checklist komen. Ik zie te vaak dat mensen maar wat doen. Als je live gaat moeten er een aantal dingen gebeuren: Welke activiteiten je gaat doen. Dat moet als eerste.. Dit is niet altijd helder. Ik zou beginnen met een checklist om af te vinken welke stappen men moet doorlopen. **Van Tongeren:** Dan maak je dus een custom checklist aan waar je gewoon handmatig punten kan toevoegen. Want de punten kunnen per klant verschillen. **Soepenberg:** Er moet een draaiboek komen om alles te documenteren voor de service & care. Dus als het eenmaal staat, hoe moet men de upgrades uitvoeren. **De Nie:** Er moeten volledige plannen komen. **Soepenberg:** Een deploymentplan inderdaad. **De Nie:** Je hebt ook de acceptatietest in deze fase. Die moet ontwikkeld en uitgevoerd worden. **Soepenberg:** En ik zou eindigen met een laatste controle. Klopt alles? Hebben we alles gedaan? Dan kan het naar Service & Care. **Van Tongeren:** Ik zou zeggen; Acceptatietest, deploymentplan en de laatste controle kritisch. **Allen:** Ja.

Vastgesteld proces en kritische processen:
Bepalen uitrolactiviteiten → Acceptatietest ontwikkelen → Deploymentplan opstellen → Upgrade draaiboek opstellen → Is alles klaar voor productie.

Versiebeheer

Versie	Activiteiten
0.1	Eerste versie scriptie
0.2	- Deelvragen toegevoegd - Interviewverwijzingen aangepast - Samenvatting toegevoegd - Voorwoord toegevoegd - Inleiding theoretisch kader toegevoegd - Theoretisch kader lopend geheel gemaakt - Theoretisch kader ingeschikt/samengevat op sommige plaatsen - Prioritering gemaakt in behandelde zaken theoretisch kader - Hoofdstuk 4 - Methode van onderzoek toegevoegd - Inleiding aangepast van hoofdstuk 5.1 - Kwaliteit en Kwaliteitsmanagement - Resultaten hoofdstuk 5.1.3 hierboven - Vergelijking Thinkwise met de literatuur toegevoegd - Resultaten hoofdstuk 5.1.4 - Definitiebepaling aangepast - Resultaten hoofdstuk 5.1.5 - Het managen van kwaliteit aangepast - Resultaten hoofdstuk 5.1.6 - Kwaliteitsmanagement bij Thinkwise herschreven - Inleiding aangepast van hoofdstuk 5.2 - Software Development Lifecycle - Resultaten hoofdstuk 5.2.1 - Definitie aangepast - Onderbouwing keuzes voor SDLC modellen in hoofdstuk 5.2.2 - Best practices volgens de literatuur - Vergelijking hoofdstuk 5.2.2 - Best practices volgens de literatuur - hantering en fasen uitgebreid en verder verdiept - Vergelijking hoofdstuk 5.2.2 - Best practices volgens de literatuur – Output en documentatie uitgebreid en verder verdiept - Conclusie hoofdstuk 5.2.2 - Best practices volgens de literatuur aangepast - Hoofdstuk 5.5 - Implementatie van de veranderingen toegevoegd. - Hoofdstuk 6 - Conclusie bijgewerkt - Hoofdstuk 7 - Aanbevelingen toegevoegd