Afstudeerverslag

"Het ontwikkelen van een offertetool"

Student:	Kevin van Dijk
Studentnummer:	08017514
Afstudeerbegeleider:	P.R.C. Breukel
2e Examinator:	A.M.J.J. Lousberg-Orbons
Opleiding:	Informatica
Bedrijf:	KBITE Automatisering
Opdrachtgever:	Remco van der Kaaij
Bedrijfsmentor:	Leon Beeloo
Datum:	01-06-2012
Versie:	1.0

Referaat

Dit verslag beschrijft het proces dat is doorlopen bij het uitvoeren van de afstudeeropdracht “Het ontwikkelen van een offertetool” bij KBITE Automatisering in ’s-Gravenzande.

Afstudeerder:	Kevin van Dijk
Onderwijsinstelling:	Haagse Hogeschool
Opleiding:	Informatica
Duur:	06-02-2012 - 01-06-2012
Organisatie:	KBITE Automatisering
Bedrijfsmentor:	Leon Beeloo
Opdrachtgever:	Remco van der Kaaij
Descriptoren:	Oracle Application Expres PL/SQL Database Offertes

Voorwoord

Dit afstudeerverslag is geschreven in het kader van mijn afstudeeropdracht die is uitgevoerd bij het bedrijf KBITE Automatisering. De opdracht betrof het ontwikkelen van offertetool.

Vanaf deze plek wil ik mijn bedrijfsmentor Leon Beeloo en opdrachtgever Remco van der Kaaij bedanken voor de tijd en moeite die ze in mij hebben gestoken. Verder wil ik de examinatoren Paul Breukel en Alwine Louseberg –Orbons bedanken voor de feedback op mijn documentatie.

Kevin van Dijk
’s-Gravenzande, 1 juni 2012

Inhoudsopgave

1. Inleiding

Dit document geeft een beschrijving over het proces dat is doorlopen tijdens het uitvoeren van mijn afstudeeropdracht. Tijdens mijn afstudeerperiode heb ik gewerkt aan het ontwikkelen van een offertetool waarmee offertes kunnen worden opgesteld aan de hand van vragen die de gebruiker krijgt.

In dit document geef ik eerst een beschrijving van de opdracht. Hierin komen de probleemomschrijving, doelstelling en resultaat aan bod.

Hoofdstuk 3 geeft een beschrijving van de organisatie waar de afstudeeropdracht is uitgevoerd en mijn plaats binnen deze organisatie. Tevens geef ik in dit hoofdstuk uitleg over Oracle Application Express. De ontwikkeltool waarmee de applicatie is gebouwd.

In hoofdstuk 4 geef ik een beschrijving van de situatie bij aanvang. Vervolgens geef ik in hoofdstuk 5 een beschrijving van de gebruikte aanpak voor dit project en de gebruikte technieken.

Hoofdstuk 6 geeft een beschrijving van de oriëntatiefase die ik aan het begin van de afstudeerperiode heb uitgevoerd. Hierin laat ik zien welke activiteiten zijn uitgevoerd en welke producten dit heeft opgeleverd om het project goed te kunnen starten.

In de hoofdstukken 7 t/m 9 beschrijf de activiteiten die zijn uitgevoerd tijdens het uitvoeren van de afstudeeropdracht aan de hand van de fases van RUP. Ik wil benadrukken dat deze iteratief zijn uitgevoerd maar om dit verslag prettig leesbaar te houden deze per fase zijn beschreven.

In hoofdstuk 10 evalueer ik de afstudeerperiode en in hoofdstuk 11 beschrijf ik welke afwijkingen er zijn ten opzichte van het opgestelde afstudeerplan.

Ten slotte komen in hoofdstuk 12 de beroepstaken aan bod.

2. Opdrachtomschrijving

2.1 Probleemomschrijving

Op dit moment kosten de offerte trajecten teveel tijd en inspanning. Vaak worden specificaties van producten/diensten meerdere malen opnieuw opgezocht om een offerte op te stellen. Om dit proces te vergemakkelijken zijn in een eerder stadium binnen de organisatie alle activiteiten en producten gestandaardiseerd en beschreven in productgroepen, producten en oplossingen. Tevens duurt het erg lang om bij een klant op locatie eerst te inventariseren wat de wensen zijn, terug naar kantoor te gaan om dit uit te werken en vervolgens weer aan de klant terug te koppelen. Dit kost klanten en KBITE teveel tijd. Daarnaast kijkt een klant veelal wat de kosten zijn en verdiept zich niet in wat hij precies koopt. Ook is de offerte mogelijkheid binnen het huidige administratie pakket te simpel en worden de offertes voor een klant (die de IT materie vaak slecht begrijpt) veelal onoverzichtelijk. Hierdoor is een tool noodzakelijk die het offertetraject verbetert, versnelt, en professionaliseert.

2.2 Doelstelling

Het doel van de opdracht is het ontwikkelen van een offertetool op basis van het Oracle APEX platform. Deze tool moet het opstellen van een offerte versnellen en professionaliseren. Met deze tool moet binnen een aantal handelingen een offerte kunnen worden opgesteld. Het systeem zal (naar verwachting) bestaan uit twee delen. Een backend waarin gegevens over producten en diensten kunnen worden bijgehouden en nieuwe producten en diensten kunnen worden toegevoegd. En een front-end; Hier zal op een wizard gestuurde manier de medewerker worden ondersteund bij het maken van de keuzes voor het opstellen van een offerte. Ook is er een idee dat dit in de toekomst kan worden uitgebreid naar direct online een offerte aanvragen. Daarnaast moet het een fool-proof systeem worden met verschillende intelligente onderdelen.

2.3 Resultaat

Het resultaat van de afstudeeropdracht zal een ontwikkelde offerte tool zijn. Met deze tool kunnen nieuwe producten worden toegevoegd en geconfigureerd, zodat het in de toekomst herbruikbaar en schaalbaar is. De medewerker zal binnen een aantal stappen een offerte kunnen opstellen. Dit zou direct in een gesprek met een klant kunnen, zodat de offerte direct kan worden opgesteld. Tevens kan de interesse en reactie van de klant worden bepaald op de uitkomst. Het zal vooral veel tijd besparen, daarnaast zorgt het voor minder kans op fouten maar ook voor professionalisering en duidelijkheid naar de klant.

3. Achtergrond

In dit hoofdstuk wordt een beschrijving gegeven van de organisatie en de plaats van de afstudeerder. Tevens wordt een beschrijving gegeven van Oracle Application Express om het ontwikkelen met deze tool beter te kunnen begrijpen.

3.1 KBITE Automatisering

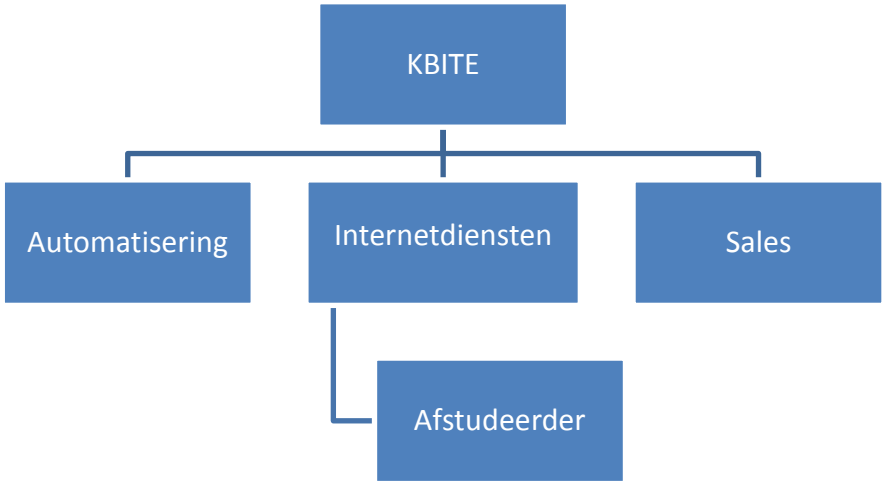
KBITE Automatisering is een ICT dienstverlener, die zich richt op het ontzorgen van haar klanten op het gebied van ICT d.m.v. verschillende ICT diensten en oplossingen. KBITE is opgericht in 2004 als eenmanszaak in systeemontwikkeling. In 2007 is KBITE Automatisering en internetdiensten ontstaan door een samengaan met van der Kaaij ICT beheer die zich richtte op kantoorautomatisering, hardware en netwerken. Op dit moment zijn er 5 personen(excl. partners) werkzaam en wordt er samengewerkt met diverse partijen. KBITE Automatisering richt zich op het MKB, en heeft meer dan 100 klanten verspreid door heel Nederland. 80% van de klanten is afkomstig uit regio Zuid-Holland. Het bedrijf is op gesplitst in twee onderdelen:

Automatisering: De opdrachten lopen uiteen van telefonie, ondersteuning en storingen, tot het inrichten, migreren en beheren van netwerken tot +/- 100 werkplekken inclusief licentiebeheer, back-up, internetverbindingen en benodigde leveringen aan hard/ en software.

Internetdiensten: De opdrachten zijn internetprojecten van ontwerp tot realisatie. Veelal zijn dit websites, webshops en maatwerk webapplicaties. Daarnaast worden er diverse integratie- en systeemkoppelingen ontwikkeld op maat voor de klant. In een samenwerkingsverband met PortalPlus is KBITE ook veel in het hogere segment actief. Hiervoor wordt software ontwikkeld veelal op basis van Oracle Application Express (APEX) en Oracle Portal.

3.2 Plaats afstudeerder

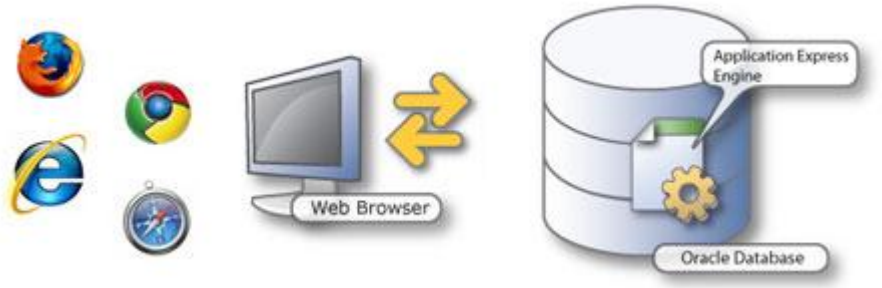
De wens van de opdracht is ontstaan vanuit de afdeling sales. Binnen de organisatie heeft de afstudeerder de opdracht uitgevoerd vanuit de afdeling internetdiensten. In figuur 1 is een organogram van de organisatie te zien met daarin de plaats van de afstudeerder.



Figuur 1 Organogram KBITE

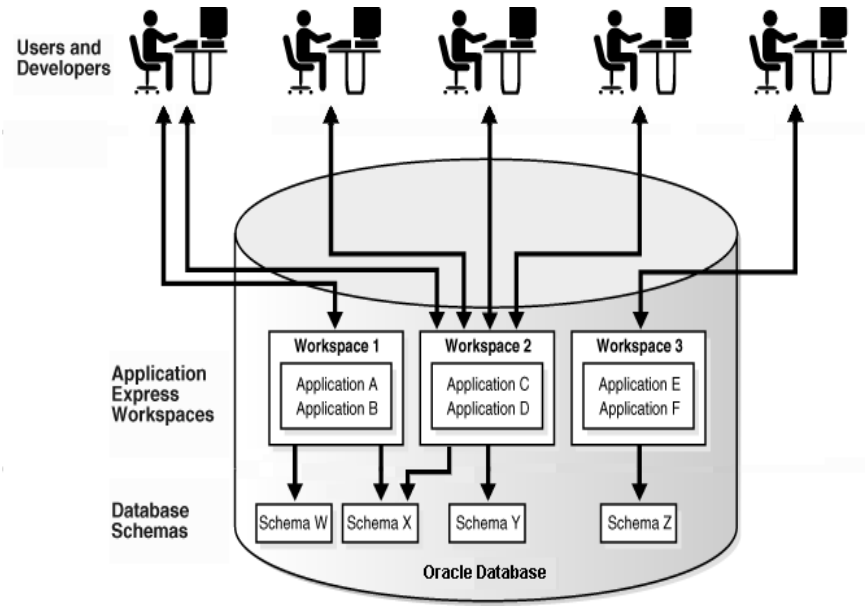
3.3 Oracle Application Express

Oracle Application Express, afgekort APEX, is een web gebaseerde ontwikkeltool waarmee webapplicaties op de Oracle database kunnen worden gemaakt. Het grote voordeel van APEX is dat het gebruikt kan worden vanuit elke browser en dat het geen aparte applicatieserver vereist. Met Oracle Apex kunnen vrij snel webapplicaties worden gemaakt door o.a. de aanwezigheid van standaard gedefinieerde invulformulieren en rapportagemogelijkheden.



Figuur 2 Oracle Application Express

In de Oracle Apex omgeving kunnen meerdere workspaces worden aangemaakt. Binnen zo'n workspace kunnen een of meerdere applicaties worden ontwikkeld, die geassocieerd zijn met een database schema. Een applicatie gemaakt met Oracle Apex bestaat uit pagina's, die opgebouwd zijn uit regions. Een region bevat content bestaande uit html tekst, page items(bv. Select list, Text Area) of region buttons. Op een pagina kunnen ook page processes worden aangemaakt die een PL/SQL functie uitvoert voor bijvoorbeeld het ophalen of toevoegen van data. Een PL/SQL functie kan direct in Oracle Apex worden geschreven of in de DBMS zodat de functie alleen moet worden aangeroepen. Dit laatste heeft als voordeel dat een functie maar 1 keer hoeft te worden gemaakt en op meerdere pagina's kan worden aangeroepen.

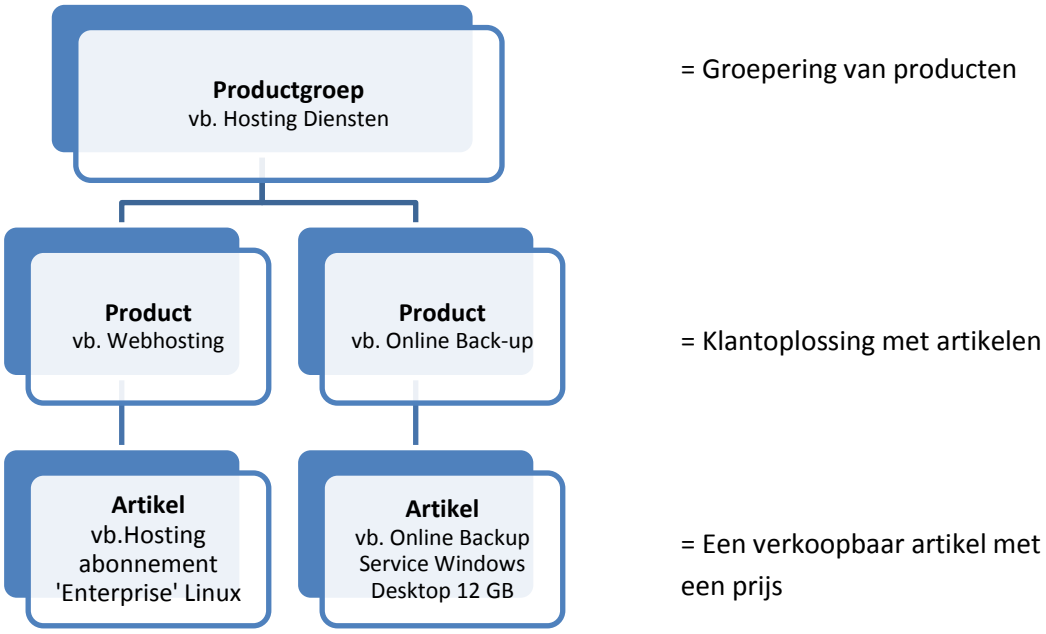


Figuur 3: illustratie Oracle Apex structuur

4. Situatie bij aanvang

Uit de opdracht is af te leiden dat men binnen de organisatie ontevreden is over de huidige gang van zaken omtrent het opstellen van offertes. Om dit proces te vergemakkelijken zijn in een eerder traject alle producten/diensten die worden geleverd verdeeld in productgroepen. Echter het opstellen van offertes met het huidige administratiepakket vergt nog steeds veel tijd en moeite. Hierdoor is binnen de organisatie besloten dat er een aparte tool moet komen die het opstellen van offertes vergemakkelijkt en versnelt.

Bij aanvang van de afstudeerperiode was er een productcatalogus aanwezig. Uit deze productcatalogus is op te maken dat de organisatie 23 producten kent die zijn verdeeld over 9 productgroepen, een voorbeeld hiervan wordt weergegeven in figuur 4. Deze indeling is een gegeven feit en wordt gebruikt ter verduidelijking richting de klant toe op o.a. reclame folders en de website. Ik noem hier deze indeling omdat in het vervolg van dit verslag hierop terug wordt gekomen. Met de kennis van de productcatalogus was duidelijk dat de offertetool van inventarisatie tot eindproduct gerealiseerd moest worden.



Figuur 4: Een gedeelte van de indeling vanuit productgroep hosting

5. Aanpak

In dit hoofdstuk wordt een beschrijving gegeven van de gekozen ontwikkelmethode voor dit project en een beschrijving geven over welke technieken zijn gebruikt.

5.1 Ontwikkelmethode

Voor het uitvoeren van dit project heb ik gekozen voor de ontwikkelmethode Rational Unified Proces(RUP). Dit is een iteratieve ontwikkelmethode voor het ontwikkelen van software. Ik heb gekozen voor de ontwikkelmethode RUP omdat mij deze het meest geschikt lijkt voor dit project. RUP sluit uitstekend aan op UML, wat binnen dit project gebruikt zal worden en ik heb al enige ervaring met RUP. Daarnaast hanteert de organisatie geen vaste ontwikkelmethode maar was er wel enige kennis over RUP. Andere populaire ontwikkelmethodes zoals XP en SCRUM heb ik in overweging genomen maar vond ik ongeschikt. SRUM wordt voornamelijk gebruikt bij projecten die door meerdere personen worden uitgevoerd, deze is dus ongeschikt omdat ik dit project alleen uitvoer. Dit zelfde geldt voor XP waarbij er gebruik wordt gemaakt van two- pair programming.

RUP deelt het project op in een aantal duidelijke fases:

- Inception fase

De inception fase is erop gericht om de inhoud, scope en haalbaarheid van het project te bepalen.

- Elaboration fase

Tijdens de elaboration fase wordt er aandacht besteed aan het ontwerp. De functionele eisen worden verder uitgewerkt tot een systeemontwerp.

- Construction fase

In de construction fase ligt de focus op het ontwikkelen van het systeem.

- Transition fase

In de transition fase wordt het eindproduct getest door de belanghebbenden. Tevens wordt er gezorgd voor overdracht en nazorg.

RUP biedt de mogelijkheid om het werk te verdelen in iteraties, dit zijn kleine deelopleveringen van het eindproduct. Door gebruik te maken van iteraties wordt het risico verkleind dat het eindresultaat niet is wat de opdrachtgever wenst. Na elke iteratie wordt er namelijk gecontroleerd of het deelproduct nog aan de eisen voldoet. Dit zorgt ervoor dat het eindproduct precies zal voldoen aan de wensen van de opdrachtgever. De opdrachtgever liet mij vrij in de keuze voor het gebruik van iteraties. Ik heb besloten om gebruik te maken van iteraties tijdens de elaboration en construction fase omdat de eisen en wensen op dat moment bekend zijn en iteraties volgens RUP hierop kunnen worden ingedeeld. Bij aanvang van deze fases is een iteratieplan opgesteld waarin is beschreven welke activiteiten worden uitgevoerd en hoeveel tijd hiervoor beschikbaar is.

5.2 Technieken

UML
Voor de ontwerpen van het systeem is gebruik gemaakt van UML.UML staat voor Unified Modeling Language en wordt gebruikt voor het maken van analyses en ontwerpen van het systeem. Dit gebeurt aan de hand van verschillende diagrammen en modellen die UML kent.

PL/SQL
PL/SQL(Procedural/Structered Query Language) is een programmeertaal en onderdeel van de Oracle-Database. Deze taal wordt gebruikt voor het schrijven van procedures/triggers en andere functies om de database te bevragen.

SQL
SQL(Structured Query Language) wordt gebruikt voor het bevragen en het aanpassen van gegevens in de database. SQL wordt in dit project dus gebruikt in combinatie met PL/SQL.

6. Oriëntatie fase

6.1 Oriëntatie & Oracle Apex onderzoek

De afstudeerperiode ben ik begonnen met het oriënteren op de opdracht. Hierbij heb ik de diverse producten en diensten, die het bedrijf levert, bestudeerd om de inhoud van de producten en diensten te kennen. Dit was nodig omdat producten veel verschillende mogelijkheden hebben. Bijvoorbeeld bij het leveren van een server zijn verschillende mogelijkheden zoals: een server met terminal services, een server met small business server of een server met exchange server. Bij het opstellen van offertes via wizard gestuurde vragen zal hier namelijk naar gevraagd moeten worden. De offertetool moet hier dus op worden afgestemd.

De eerste 2 weken van het afstudeertraject heb ik ook gebruikt om meer kennis op te doen van Oracle Apex. Hiervoor heb ik diverse tutorials gevolgd die door Oracle beschikbaar worden gesteld. In deze tutorials wordt geleerd hoe een applicatie met Oracle Apex i.c.m. een Oracle database 11g moet worden gemaakt. Na het volgen van deze tutorials is er overleg geweest met de bedrijfsmentor. In dit overleg is de opgedane kennis besproken en is er besproken welke functionaliteiten nog verder moeten worden onderzocht. Deze heb ik vervolgens onderzocht door het maken van een kleine testapplicatie. Alle opgedane kennis heb ik bijgehouden in een document zodat ik hierop terug kan vallen tijdens het bouwen van de offertetool.

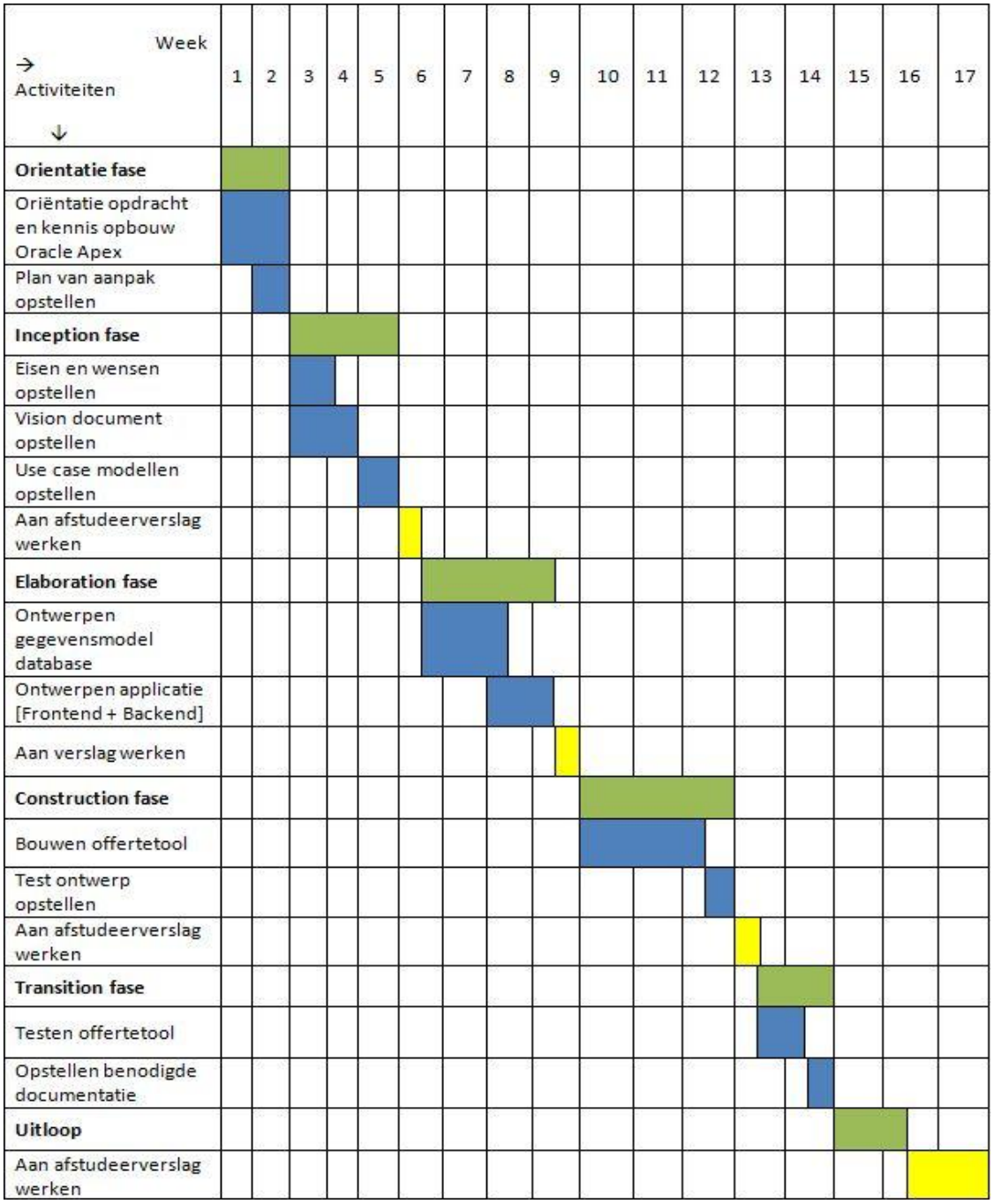
6.2 Opstellen Plan van Aanpak

In de tweede week van het afstudeertraject heb ik het plan van aanpak opgesteld. Door dit in de tweede week van de oriëntatiefase te doen konden activiteiten, methodes, technieken en tools beter worden bepaald. Om een zo compleet mogelijk document te krijgen heb ik het plan van aanpak meerdere malen besproken met de opdrachtgever. Belangrijke punten van beide kanten konden zo besproken worden, wat uiteindelijk heeft geresulteerd in een document dat voor beide partijen duidelijk was. Het plan van aanpak geeft een beschrijving over de aanpak van dit project en dient als leidraad voor de uitvoering van het project.

6.2.1 Planning

Een belangrijk onderdeel van een project is het maken van een planning. Voor aanvang van het project is al een afstudeerplan opgesteld waarin activiteiten zijn beschreven met een globale tijdsschatting. Dit document is gebruikt als input om de planning op te stellen. De planning is opgedeeld aan de hand van de fases die RUP beschrijft. Hierbij is voorafgaand aan de RUP fases een oriëntatiefase toegevoegd. Deze fase is bedoeld om meer duidelijkheid over de opdracht te krijgen en kennis op te doen van Oracle Apex. Om vertragingen tijdens het project op te vangen is in de planning een uitloop fase opgenomen.

In figuur 5 is de planning weergegeven die aan het begin van het project is opgesteld. Deze planning doet misschien denken aan een waterval uitvoering maar dit is een vertekend beeld en moet eigenlijk worden gezien om de activiteiten te verdelen over de beschikbare tijdsperiode. Zoals eerder verteld heb ik voor aanvang van de elaboration en construction fase een iteratieplan opgesteld waarin deze activiteiten specifiek per iteratie staan beschreven. Deze iteratieplannen zijn in de bijlage terug te vinden.



Figuur 5: Planning

6.2.2 Op te leveren producten

Bij aanvang van het project is afgesproken om de volgende producten op te leveren:

- Plan van Aanpak
- Vision document
    - Eisen en wensen offertetool
    - Haalbaarheid, belanghebbende en behoeftes
- Use Case Model document
    - Use Case Diagram
    - Use Case beschrijvingen
- Ontwerp gegevensmodel database
    - Klassendiagram
    - Relationeel Representatie model
    - Relationeel Implementatie model
- Ontwerp applicatie Frontend & Backend
    - Functioneel/technisch ontwerp applicatie
- Testontwerp
- Testresultaten
- Benodigde documentatie
- Eindresultaat: Offertetool

7. Inception fase

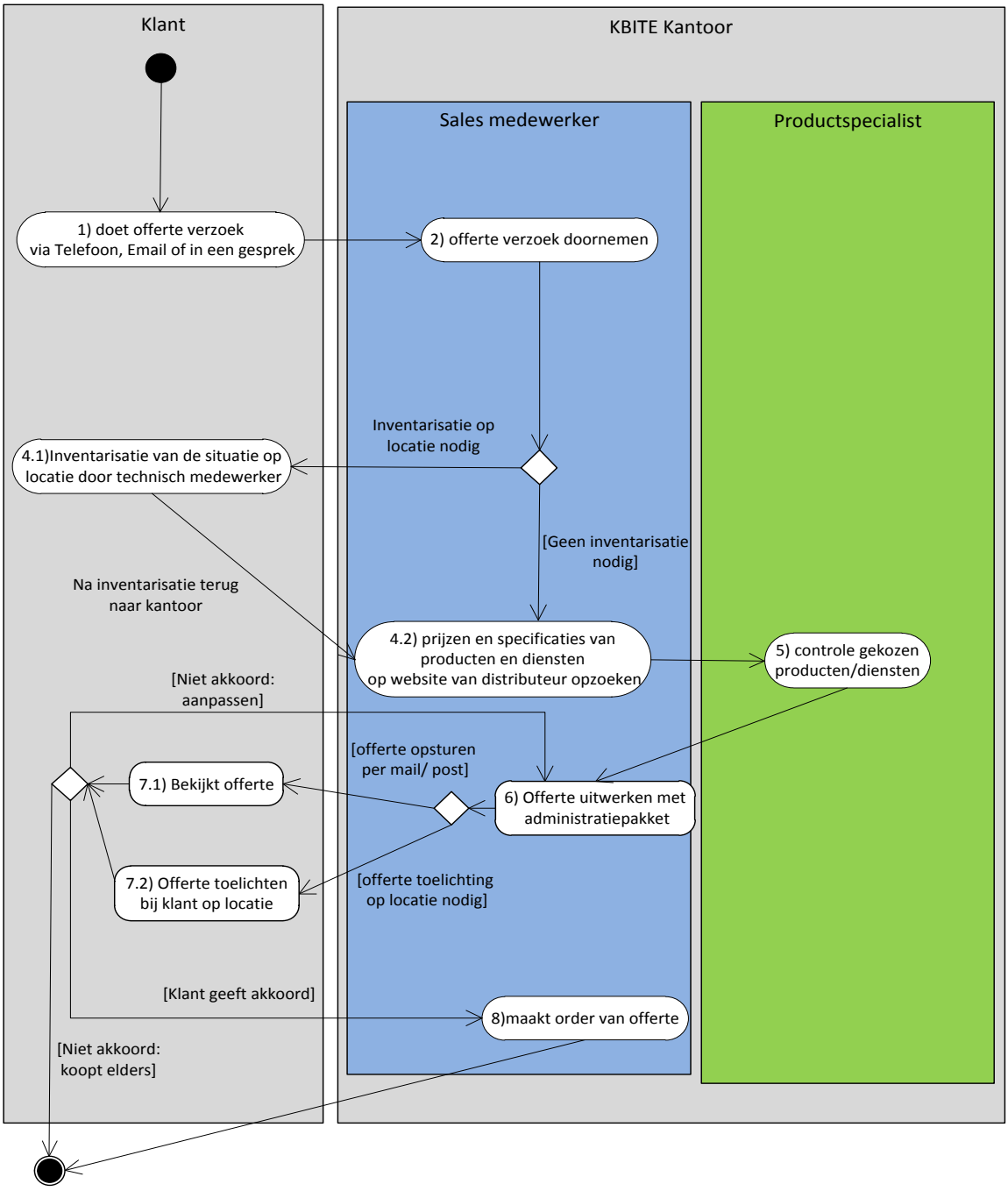
De inception fase is de eerste fase die de RUP ontwikkelmethode beschrijft. In deze fase wordt de inhoud, scope en haalbaarheid van het project bepaald.

Tijdens dit project zijn in de Inception fase de volgende activiteiten verricht:

- Vision Document opstellen
- Eisen en wensen opstellen
- Use Case Document opstellen

7.1 Vision document opstellen

Om de gezamenlijke visie over het project tussen opdrachtgever en opdrachtnemer vast te leggen is het Vision document opgesteld. Voor dit document heb ik gebruik gemaakt van een template die beschikbaar wordt gesteld door RupOpMaat. Om de visie duidelijk te krijgen heb ik meerdere gesprekken gevoerd met de opdrachtgever en heb ik mee kunnen kijken hoe offertes met het huidige administratie programma worden opgesteld. Dit heeft mij inzicht gegeven om het huidige bedrijfsproces in kaart te brengen. Door dit bedrijfsproces, weergegeven in figuur 6, in kaart te brengen wordt duidelijk welke activiteiten allemaal moeten worden uitgevoerd binnen het offertetraject. Hier blijkt o.a. uit dat er veel tijd wordt besteed aan uitzoekwerk en communicatie met de klant en tussen interne werknemers. Het eindresultaat van dit project zal deze punten dus aanzienlijk moeten verbeteren. De opdrachtgever was erg tevreden over de visuele weergave van het bedrijfsproces. Op deze manier werden ook voor hem de knelpunten duidelijker.



Figuur 6 Huidig bedrijfsproces opstellen offertes

Om te zorgen dat het eindproduct aansluit bij de wensen van de eindgebruikers heb ik de eindgebruikers eerst in kaart gebracht. Deze hebben namelijk verschillende verantwoordelijkheden met de offertetool. Bij het ontwerpen en bouwen moet hiermee rekening worden gehouden. Met het huidige administratieprogramma kunnen gebruikers geen verschillende verantwoordelijkheden krijgen. Men kan het administratieprogramma gebruiken of niet. Om in het eindresultaat wel een duidelijke scheiding tussen verantwoordelijkheden te creëren heb ik bijbehorende systeemrollen bedacht. Deze heb ik vervolgens voorgelegd aan de opdrachtgever om te bepalen of deze voldoende zijn. Hier kwam direct de vraag naar boven over in hoe verre de scheiding tussen verantwoordelijkheden door moet gaan. Bijvoorbeeld of er voor

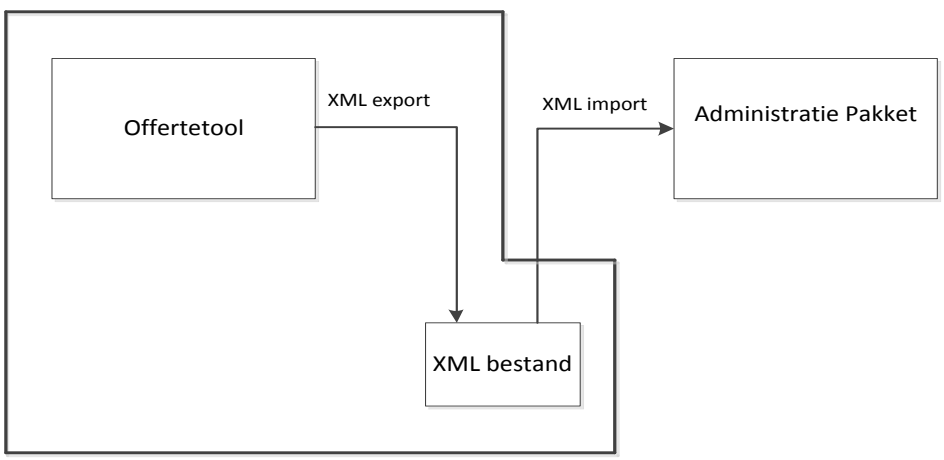
het beheren van de productgegevens een aparte rol moet komen voor het bekijken van inkoopprijzen en een aparte rol voor het bekijken van verkoopprijzen. Ik heb voorgesteld om de belanghebbende rollen met het systeem in eerste instantie zo simpel mogelijk te houden. Anders ontstaat er voor elke kleine functie een rol en zijn er uiteindelijk meer rollen dan personen binnen de organisatie. Als er in de toekomst nieuwe rollen bijkomen dan kunnen deze met Oracle Apex gemakkelijk worden uitgebreid. De opdrachtgever vond dit een goede benadering waarna bijbehorende personen aan de rollen zijn gekoppeld. Hieronder is een beschrijving weergegeven van de opgestelde rollen met hun verantwoordelijkheden. In een later stadium, o.a. bij het opstellen van use cases, zijn de verantwoordlijkheden verder gespecificeerd.

Sales manager
De sales manager zal binnen de offertetool alle functionaliteiten kunnen benutten. Echter de voornaamste verantwoordelijkheid met de offertetool zal het beheren zijn van alle offertes die door de sales medewerkers zijn gemaakt en ook het kunnen opstellen van offertes.

Sales medewerker
De sales medewerker zal verantwoordelijk zijn voor het opstellen van offertes en zal de mogelijkheid hebben voor het bekijken, wijzigen en verwijderen van zijn eigen gemaakte offertes.

Productspecialist
De productspecialist zal verantwoordelijk zijn voor het beheren van artikelgegevens, producten en voor het instellen van de offertetool zodoende dat er offertes mee kunnen worden opgesteld.

Een ander aspect binnen de vision is het productperspectief. Voor mij was namelijk niet duidelijk of de offertetool het huidige administratieprogramma moest vervangen of dat ze naast elkaar moesten gaan werken. In een gesprek met mijn begeleider en opdrachtgever werd mij verteld dat de offertetool de offertefunctie van het administratiepakket op termijn moet gaan vervangen en dat het administratiepakket een XML import mogelijkheid heeft. Gemaakte offertes kunnen zodoende binnen het administratieprogramma worden geïmporteerd om te verwerken als order en verder administratief worden afgehandeld. De visuele weergave van het productperspectief is te zien in figuur 7.



Figuur 7 Productperspectief

Het vision document is meerdere malen aangepast om tot een definitieve versie te komen. Op de definitieve versie heeft de opdrachtgever akkoord gegeven. Dit betekent dat hij het eens is over de inhoud en deze gebruikt kan worden om op terug te vallen tijdens het project als er bijvoorbeeld discussie ontstaat over de eisen.

7.2 Eisen en wensen opstellen

Een belangrijk onderdeel in de inception fase is het opstellen van de eisen en wensen. Om deze zo goed mogelijk op papier te krijgen heb ik meerdere interviews gevoerd met de opdrachtgever. In deze interviews heb ik gevraagd wat de grootste knelpunten zijn waarom de offertetool er moet komen en welke specifieke functionaliteiten het eindproduct moet hebben. Deze zijn belangrijk om als focus te houden en bij het testen van het product. Tevens heb ik eisen en wensen kunnen afleiden uit het in kaart brengen van het huidige bedrijfsproces. Uit de gesprekken met de opdrachtgever en uit het huidige bedrijfsproces zijn punten naar voren gekomen die moeten worden verbeterd, deze zijn:

Offertetraject verkorten

In het huidige bedrijfsproces duurt het gehele offertetraject te lang doordat producten en diensten elke keer moeten worden opgezocht of moeten worden opgevraagd bij een technische medewerker. Tevens is er geen mogelijkheid om op locatie bij de klant een offerte op te stellen wat ervoor zorgt dat een medewerker eerst terug naar kantoor moet om daar de offerte op te stellen. Vervolgens moet dit weer gecommuniceerd worden met de klant.

Verkorten van tijdsinspanning en moeite

Het opstellen van een offerte kost te veel tijd en moeite doordat in een lange lijst met artikelen het juiste artikel moet worden gezocht om aan en offerte toe te kunnen voegen. Tevens moet de website van de distributeur worden gecontroleerd of de prijs en specificaties nog up to date zijn.

Verduidelijken van offertes

Binnen de organisatie is men ontevreden over de huidige uitvoering van offertes. Deze komen onduidelijk over naar de klant vanwege het ontbreken van een toelichting op gekozen artikelen.

Klant beter betrekken in offertetraject

De klant wordt op dit moment te weinig betrokken in het offertetraject. De klant maakt kenbaar wat hij wil en de sales medewerker werk een offerte hiervoor uit. De klant weet hierdoor niet welke gevolgen de gemaakte keuzes hebben op de kosten. Hierdoor ontstaat de kans dat de klant ingaat op een goedkopere offerte van de concurrent terwijl deze een totaal andere oplossing aanbiedt.

Geen technische kennis nodig voor opstellen offertes

Op dit moment moeten sales medewerkers bij het opstellen van een offerte technische informatie over producten/diensten opvragen bij de technische medewerkers. Dit komt omdat sales medewerkers geen kennis hebben of producten/diensten technisch geschikt zijn voor bepaalde oplossingen.

Om te zorgen dat er geen belangrijke kwaliteitseisen aan het eindproduct worden vergeten is er gebruik gemaakt van het ISO 9126 model. Dit model beschrijft de kwaliteitskenmerken van software aan de hand van de volgende categorieën:

- Functionaliteit
- Betrouwbaarheid
- Bruikbaarheid
- Efficiëntie
- Onderhoudbaarheid
- Overdraagbaarheid

De functionaliteitseisen zijn verder ingedeeld aan de hand van de onderwerpen die aanbod komen in de offertetool. Om onduidelijkheden te voorkomen zijn alle eisen zo specifiek mogelijk uitgewerkt, in hoeverre die op dat moment bekend waren. Ik heb de eisen vervolgens geprioriteerd die naar mijn inziens het belangrijkste zijn. Het prioriteren heb ik gedaan aan de hand van de MoSCoW-methode.

De letters in deze afkorting staan voor:

- **M**ust have
  Deze eis moet in het systeem terugkomen
- **S**hould have
  Deze eis is zeer gewenst voor het systeem maar het systeem kan wel functioneren zonder deze eis
- **C**ould have
  Als er genoeg tijd over is komt deze eis aan bod
- **W**on't have
  Deze eis zal tijdens dit project niet aan bod komen maar kan interessant zijn voor in de toekomst

In een bespreking heb ik de lijst met eisen voorgelegd aan de opdrachtgever om zijn mening hierover te pijlen. Punten die de opdrachtgever zeker terug wil zien zijn vervolgens gemarkeerd en minder belangrijke punten mocht ik zelf een prioriteit aangeven. Dit heb ik gedaan aan de hand van een tijdsschatting wat haalbaar moest zijn voor mijn afstudeerperiode. Voor aanvang van het project had ik gedacht om een apart document voor de eisen en wensen op te stellen. Echter bij het zoeken naar extra informatie over RUP ben ik o.a. het eerder genoemde Vision template van RupOpMaat tegen gekomen. Hierin worden eisen en wensen beschreven onder behoeftes en producteigenschappen. Ik heb toen besloten om dit aan te houden zodat er 1 document is waarin alle punten tussen de opdrachtgever en opdrachtnemer staan vermeld.

In onderstaand overzicht is een gedeelte van de opgestelde eisen opgenomen. Voor de complete lijst met eisen verwijs ik naar het Vision document in de bijlage.

Functionaliteit

Functionele eisen m.b.t offertes		
Nr	**Eis**	**Prioriteit**
OFT 1	Er moeten gemaakte offertes kunnen worden beheerd	M
OFT 2	Er moeten offertes kunnen worden opgesteld	M
OFT 3	Aan de hand van wizard gestuurde vragen een offerte kunnen opstellen	M
OFT 4	Aan de hand van offerte template een offerte kunnen opstellen	W
OFT 5	Handmatig een offerte kunnen samenstellen	W
OFT 6	Een offerte kunnen opslaan in de database	M
OFT 7	Een offerte kunnen wijzigen	M
OFT 8	Een offerte kunnen verwijderen	M
OFT 9	Direct vanuit het systeem de offerte naar email kunnen verzenden	M
OFT 10	Een overzicht van alle gemaakte offertes kunnen bekijken	M
OFT 11	Een overzicht van eigen gemaakte offertes kunnen genereren	M
OFT 12	De status(bv. in afwachting, in behandeling, afgerond) van een offerte kunnen wijzigen	M
OFT 13	Een gemaakte offerte kunnen kopiëren naar een andere klant/groep klanten	S
OFT 15	Een gewijzigde offerte kan alleen worden opgeslagen naar een nieuwe versie	M
OFT 16	Standaard gedefinieerde documenten aan een offerte kunnen toevoegen	M
OFT 17	Een opgestelde offerte bevat altijd een bijlage met algemene voorwaarden per product	M
OFT 18	Een offerte moet meerdere artikelen kunnen bevatten	M

Functionele eisen m.b.t. productgroepen		
Nr	**Eis**	**Prioriteit**
PG 1	Er moeten productgroepen kunnen worden beheerd	M
PG 2	Een productgroep kunnen toevoegen	M
PG 3	Een productgroep kunnen wijzigen	M
PG 4	Een productgroep kunnen verwijderen	M

Functionele eisen m.b.t. producten/diensten		
Nr	**Eis**	**Prioriteit**
PD 1	Producten/diensten moeten kunnen worden beheerd	M
PD 2	Losse producten en diensten kunnen opzoeken(snel zoeken)	M
PD 3	Een product/dienst kunnen toevoegen	M
PD 4	Een product/dienst kunnen wijzigen	M
PD 5	Een producten/dienst kunnen verwijderen	M
PD 6	Gerelateerde producten/diensten kunnen aangeven	M

Functionele eisen m.b.t. artikelen		
Nr	**Eis**	**Prioriteit**
AT 1	Artikelen moeten kunnen worden beheerd	M
AT 2	Een artikel kunnen toevoegen	M
AT 3	Een artikel kunnen wijzigen	M
AT 4	Een artikelen kunnen verwijderen	M
AT 5	Losse artikelen kunnen opzoeken	M
AT 6	Er moeten documenten(bv. Specificaties) bij een artikel kunnen worden toegevoegd	M
AT 7	Er moet een foto kunnen worden toegevoegd bij een artikel	M
AT 8	Arbeidskosten voor installatie/configuratie van een artikelkunnen toevoegen	M

Functionele eisen m.b.t. wizard vragen		
Nr	**Eis**	**Prioriteit**
WV 1	Wizard vragen toe kunnen voegen	M
WV 2	Wizard vragen kunnen wijzigen	M
WV 3	Wizard vragen kunnen verwijderen	M

Functionele eisen m.b.t. het wizard proces		
Nr	**Eis**	**Prioriteit**
WP 1	Er moeten zo weinig mogelijk wizard vragen worden gesteld voor het opstellen van een offerte	M
WP 2	Het systeem mag geen overbodige vragen stellen tijdens het wizard proces	M
WP 3	Het moet mogelijk zijn om tijdens het opstellen van een offerte een stap terug te kunnen	M
WP 4	Alle ingevulde gegevens moeten tijdens het wizard proces worden gevalideerd	M
WP 5	Bij elke stap vooruit en achteruit in het wizard proces moet worden gevalideerd of de gemaakte keuzes nog kloppen	M
WP 6	Het systeem moet zo veel mogelijk ingevulde gegevens onthouden bij het wijzigen tijdens het wizard proces	M
WP 8	Arbeidskosten voor installatie/configuratie worden tijdens het wizard proces automatisch berekend op basis van de gekozen artikelen	C

Export eisen		
Nr	**Eis**	**Prioriteit**
EP 1	Het systeem moet een XML bestand kunnen genereren van de offertes	S
EP 2	Het systeem moet een PDF bestand kunnen genereren van een offerte	M
EP 4	Het systeem moet een Word document kunnen genereren van een offerte	W

Autorisatie		
Nr	**Eis**	**Prioriteit**
AT 1	Het systeem moet een mogelijkheid hebben voor gebruikersbeheer	M
AT 2	Gebruikers van het systeem moeten verschillende rechten kunnen hebben	M

Betrouwbaarheid

Nr	Eis	Prioriteit
BWB 1	De applicatie moet op geen enkel moment andere applicaties verstoren	M
BWB 2	Rekenfouten binnen de applicatie moeten worden uitgesloten	M
BWB 3	Er moet een reservekopie van het systeem worden gemaakt om gegevens verlies te vermijden	M
BWB 4	Het systeem moet makkelijk kunnen worden hersteld bij calamiteiten	M

Bruikbaarheid

Nr	Eis	Prioriteit
BKB 1	De applicatie moet een duidelijke en simpele grafische interface hebben	M
BKB 2	Bij een fout moet de applicatie een duidelijke melding naar de gebruiker geven	M
BKB 3	Het systeem moet te bereiken zijn met de laatste versies van de populaire desktop browsers(Firefox, Chrome, Internet Explorer)	M
BKB 4	Het systeem moet te bereiken zijn vanaf een Ipad	M
BKB 5	Het systeem moet te bereiken zijn vanaf een Android tablet	W

Efficiëntie

Nr	Eis	Prioriteit
EF 1	Het systeem mag geen overmatig gebruik maken van resources(cpu, netwerk, geheugen)	M

Onderhoudbaarheid

Nr	Eis	Prioriteit
OHB 1	Het systeem moet makkelijk te onderhouden zijn	M
OHB 2	Het systeem moet uitbreidbaar zijn met nieuwe functies	M

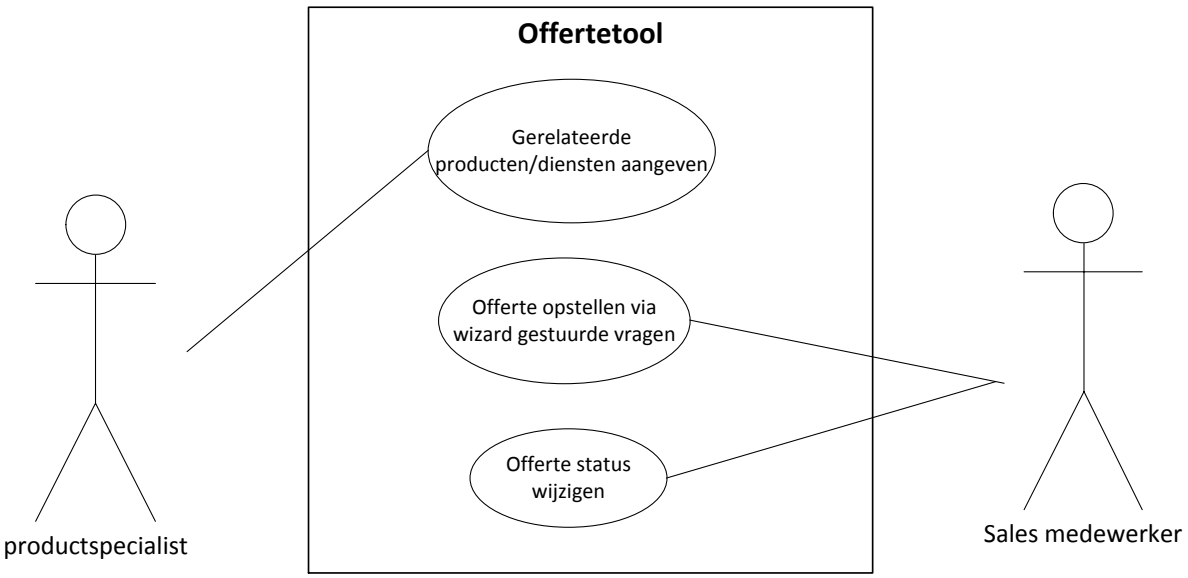
Overdraagbaarheid

Nr	Eis	Prioriteit
ODB 1	Het systeem moet worden gerealiseerd op basis van Oracle Apex	M

7.3 Use Case Modellen opstellen

Om de interactie tussen de gebruikers en het systeem weer te geven heb ik Use Case diagrammen en Use Case beschrijvingen opgesteld. Allereerst heb ik de actors opgesomd, dit zijn eigenlijk de belanghebbenden vanuit het vision document. Hier heb ik een actor aan toegevoegd; de public user. Een wens van de opdrachtgever is namelijk dat in de toekomst iedereen een offerte kan aanvragen via de website. Door hier alvast rekening mee te houden kan het systeem in de toekomst gemakkelijk worden uitgebreid.

In eerste instantie had ik een groot Use Case diagram gemaakt. Echter in het eerste gesprek met de opdrachtgever werd al duidelijk dat deze voor hem onbegrijpelijk was en heb ik deze daarom uitgesplitst in een Use Case diagram per actor. Bij het opstellen van deze diagrammen ontstond wat verwarring doordat ik me te veel aan de termen uit het afstudeerplan wilde houden. Hierin had ik namelijk geschreven over een front-end en backend applicatie. Waarbij de front-end is bedoeld om offertes op te stellen en de backend om producten en diensten te beheren. Echter als een medewerker is ingelogd via de front-end moet deze ook offertes kunnen opstellen, waardoor de front-end en backend al door elkaar lopen. Mijn bedrijfsmentor heeft mij toen het advies gegeven om bij het opstellen van de Use Case diagrammen nog geen rekening te houden met een front-end en backend applicatie omdat dit een technische invulling is en geen functionaliteit. De Use Cases en beschrijvingen kon ik hierna redelijk gemakkelijk opstellen omdat de eisen in het vision document al specifiek waren uitgewerkt. In figuur 8 is een gedeelte van de Use Case diagrammen van Productspecialist en Sales medewerker te zien. Een uitgewerkte Use case beschrijving is weergegeven in figuur 9. Voor alle Use Case Diagrammen en beschrijvingen verwijs ik naar het Use Case model document in de bijlage.



Figuur 8 Use Case Diagram Sales medewerker & productspecialist

Naam	Offerte status wijzigen
Actor	Sales medewerker, Sales manager
Preconditie	De actor heeft een offerte opgesteld.
Scenario beschrijving	1. De actor opent het offerteoverzicht 2. Het systeem toont het scherm met gemaakte offertes 3. De actor selecteert een offerte waarvan hij de status wilt wijzigen 4. Het systeem toont de offerte met de offerte status 5. De actor wijzigt de status en bevestigd 6. Het systeem slaat de gegevens op 7. Het systeem toont offerteoverzicht
Uitzonderingen	Geen
Postconditie	De status van een offerte is gewijzigd en het systeem is terug gekeerd naar het offerteoverzicht.
Requirement	**OFT 12** De status(bv. in afwachting, in behandeling, afgerond) van een offerte kunnen wijzigen

Figuur 9: Use Case beschrijving offerte status wijzigen

8. Elaboration fase

Na het uitvoeren van de inception fase was er genoeg informatie verzameld om te starten met de elaboration fase. In dit hoofdstuk zal ik een beschrijving geven van het nieuwe bedrijfsproces dat ontstaat door het gebruik van de offertetool. Daarnaast zal ik een beschrijving geven van de activiteiten die zijn uitgevoerd in deze fase. Voor deze fase heb ik gebruik gemaakt van 3 iteraties waarin de volgende activiteiten zijn verricht:

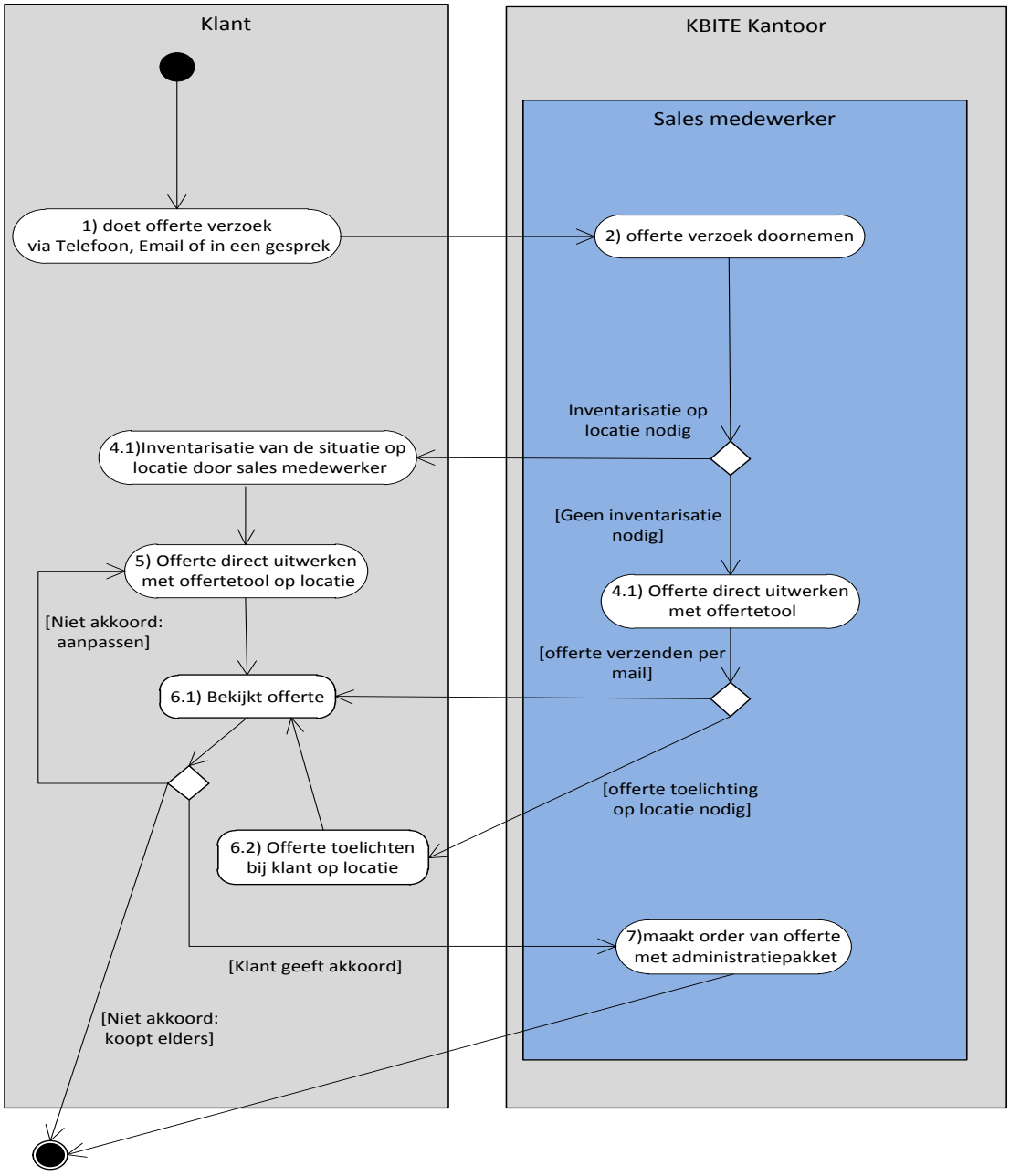
- Functioneel ontwerp applicatie maken
- Gegevensmodel voor de database opstellen
- Relationeel Representatie model opstellen
- Activitydiagram opstellen
- Relationeel implementatie model maken

8.1 Het nieuwe bedrijfsproces

Na het opstellen van de eisen en wensen en het voeren van gesprekken met de opdrachtgever is een duidelijk beeld ontstaan van het nieuwe bedrijfsproces met de offertetool. In het nieuwe bedrijfsproces zal de offertetool een grote bijdrage leveren aan het verbeteren van de genoemde punten uit de inception fase, deze waren:

- Offertetraject verkorten
- Geen technische kennis nodig voor opstellen offertes
- Klant beter betrekken in offertetraject
- Verkorten tijdsinspanning en moeite
- Verduidelijken van offertes

De offertetool zal het offertetraject verkorten door een aantal voordelen die de tool met zich mee brengt. Met de offertetool kunnen offertes direct op locatie worden opgesteld wat een besparing in reistijd tussen klant en kantoor oplevert. Tevens is voor het opstellen van offertes geen communicatie meer nodig tussen sales medewerker en productspecialist omdat alle technische kennis in de tool is verwerkt, wat ook een tijdsbesparing oplevert. Doordat offertes op locatie bij de klant kunnen worden opgesteld wordt de klant meer betrokken in het offertetraject. De klant ziet namelijk welke keuzes door de sales medewerker worden gemaakt bij het opstellen van offertes en kan direct zijn mening hierover geven. Het verkorten van tijdsinspanning en moeite komt voornamelijk doordat prijzen en specificaties niet meer hoeven worden uitgezocht maar de offertetool zal d.m.v. de wizard vragen automatisch de juiste artikelen tonen. Omdat de offertetool van begin tot eind wordt gerealiseerd kan de lay-out van offertes naar eigen wens worden bepaald en hoeft er niet aan een standaard lay-out van een gekocht softwarepakket worden gehouden. Dit zal het verduidelijken van offertes enorm verbeteren. In figuur 10 is het nieuwe bedrijfsproces visueel weergegeven.



Figuur 10: Het nieuwe bedrijfsproces

8.2 Gegevensmodel voor de database opstellen

In deze paragraaf zal ik beschrijven welke knelpunten ik heb opgelost bij het opstellen van het gegevensmodel en de keuzes die hierbij horen. Deze keuzes zijn van invloed geweest op de huidige productindeling. Om dit hoofdstuk goed leesbaar en begrijpelijk te houden verwijs ik daarom naar de huidige productindeling die is weergegeven in hoofdstuk 4.

Voorafgaand aan het opstellen van het gegevensmodel heb ik een keuze moeten over welk model ik hiervoor zou moeten gebruiken. Tijdens school projecten heb ik al vaker een gegevensmodel op moeten stellen. Hiermee heb ik ervaring opgedaan met het gebruik van een klassendiagram en een Enchanced Entity-Relationschip Model(EER model). Waarbij een klassendiagram hoofdzakelijk wordt gebruikt om de structuur van een applicatie vast te leggen maar deze ook gebruikt wordt als gegevensmodel. Het EER model wordt veelal gebruikt bij complexe databases op hoog niveau. Om hierin een juiste keuze te maken heb ik informatie gezocht en met mijn bedrijfsmentor gesproken over welk model ik het beste kon gebruiken. Hieruit bleek dat de organisatie voor dit soort projecten vrijwel altijd het klassendiagram gebruikt. Dit vanwege de algemeen bekende en duidelijke notatie, ook richting opdrachtgevers toe. Mijn persoonlijke voorkeur ging ook naar dit model uit omdat ik de notatie van het EER model onduidelijk vind en onprettig om mee te werken.

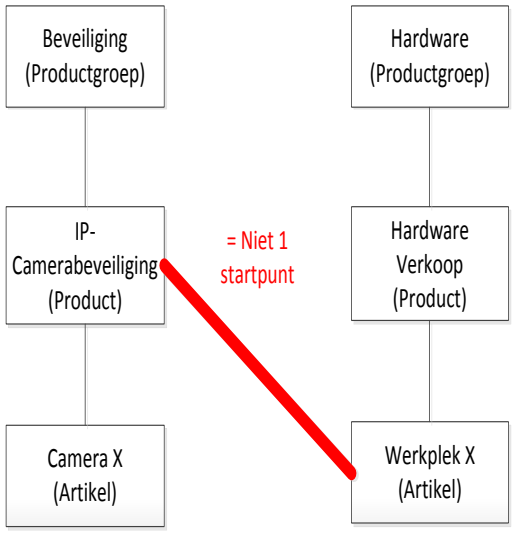
De belangrijkste eis voor het eindproduct is het opstellen van offertes aan de hand van wizard gestuurde vragen. Om dit proces duidelijk te krijgen heb ik een brainstorm sessie gehad met mijn opdrachtgever waarin dit proces in een aantal scenario's is uitgeschreven. Een van de eerste problemen die hieruit naar voren kwam was het startpunt van het wizardproces.

Het startpunt van wizardproces

In gesprek met de opdrachtgever vroeg ik hem waarvan een offerte moet worden gemaakt. Kijken naar de huidige productindeling waren er immers 3 mogelijkheden: een offerte maken van een artikel, product of productgroep. De opdrachtgever heeft mij toen enkele voorbeelden gegeven waarvan hij een offerte zou willen opstellen, deze voorbeelden waren:

- IP-Camerabeveiliging met werkplek als opslag
- Server met werkplek(ken)
- Server met terminalserver

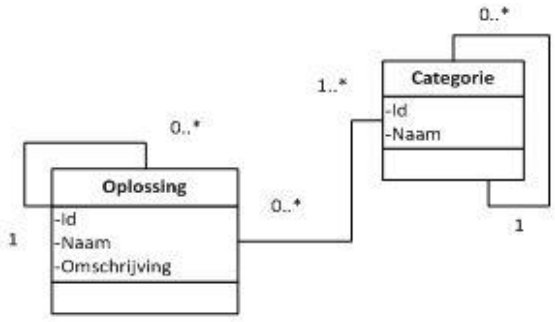
De opdrachtgever erkende dat dit geen op zichzelf staande producten zijn maar combinaties van producten en artikelen De hierboven genoemde “IP-Camerabeveiliging met werkplek als opslag” komt helemaal niet voor als 1 product in de productcatalogus. IP-Camerabeveiliging is een product uit de productgroep beveiliging en een werkplek is een artikel die voortkomt uit de productgroep Hardware. Doordat de opdrachtgever deze inconsequente combinatie van een product(IP-Camerabeveiliging) met een artikel(Werkplek) wil maken is er geen eenduidig startpunt van het wizardproces.



Figuur 11 Illustratie niet 1 startpunt

Omdat met de samengestelde offertes het meeste tijd verloren gaat moest hiervoor wel een oplossing worden bedacht. Het was aan mij de taak om hiervoor wel een duidelijk startpunt te creëren. Ik stelde voor om te starten met een “offerte oplossing”. Een oplossing kan dan naar eigen wens worden geconfigureerd en zodoende zelf de combinaties bepalen. Door dit duidelijk aan hem toe te lichten en ook visueel te laten zien in het functioneel ontwerp, die in het volgende hoofdstuk wordt beschreven, heb ik hem overtuigd om een oplossing als startpunt te houden. Om de huidige productindeling(Productgroep -> Product) richting de klanten toe intact te houden kan een oplossing als ”IP-Camera met werkplek als opslag” zowel onder het product IP-Camerabeveiliging als Hardware worden weergegeven.

In het klassendiagram heb ik dit opgelost door een veel op veel relatie van oplossing met categorie. Een categorie is eigenlijk een interne naam voor zowel productgroep als product en heeft een parent-child relatie zodat product onder Productgroep kan worden aangegeven. Op deze manier blijf ik voldoen aan de eisen van de opdrachtgever over het toevoegen, wijzigen, verwijderen van producten en productgroepen. Omdat bij het opstellen van de eisen was gedacht dat een product het startpunt zou zijn is daar de eis gerelateerde producten aangeven opgenomen maar dat is dus oplossing geworden. Door ook bij oplossing een parent-child relatie te creëren blijf ik hier aan voldoen en kunnen aan een oplossing gerelateerde andere oplossingen worden gekoppeld.



Figuur 12 Klassendiagram gedeelte oplossing - categorie

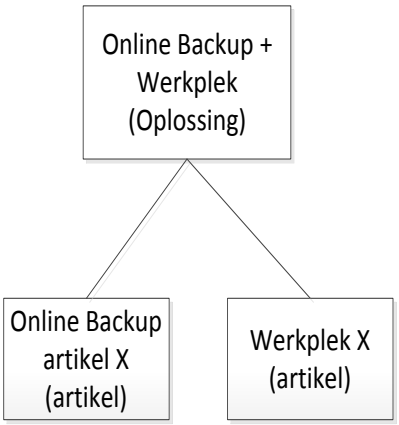
Overbodige vragen overslaan

De situatie die nu is ontstaan, weergegeven in figuur 13, is dus een oplossing die wordt gestart om tot een artikel uit te komen. Maar op deze manier was het nog niet haalbaar om te voldoen aan een aantal eisen van de opdrachtgever:

- Er moeten zo weinig mogelijk wizard vragen worden gesteld voor het opstellen van een offerte
- Het systeem mag geen overbodige vragen stellen tijdens het wizard proces

Om beter te begrijpen waarom dit niet haalbaar is zal ik hier een aantal voorbeeld vragen noemen die bij deze oplossing zouden kunnen horen om tot een artikel te komen:

- Wilt u desktop of server backupen?
- Hoeveel data wilt u backuppen?
- Wat voor processor wilt u?
- Hoeveel opslag capaciteit wilt u?



Figuur 13 Foutieve situatie

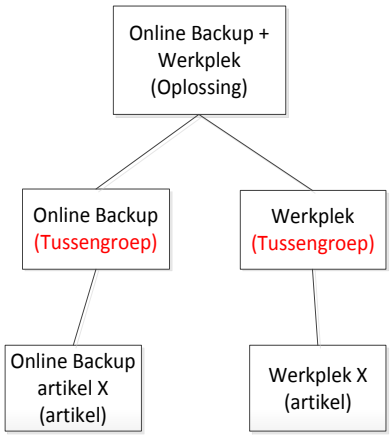
In deze situatie zou het betekenen dat alle bovengenoemde vragen zouden moeten worden gesteld om bijvoorbeeld tot “Online Backup artikel X” te komen. Terwijl de vragen “Wat voor processor wilt u”? en “Hoeveel opslag capaciteit wilt u?” om tot het artikel “Online Backup artikel x” te komen overbodig zijn. Dit zou ik kunnen verbeteren door bijvoorbeeld prioriteiten aan deze vragen te koppelen en deze uit te programmeren om vragen over te kunnen slaan. Voor 1 oplossing zou dit wel haalbaar zijn maar niet voor alle oplossingen die kunnen worden gemaakt met 23 producten die de organisatie kent. Tevens zou het systeem op deze manier niet makkelijk te beheren zijn, wat een eis is van de opdrachtgever, omdat elke nieuwe oplossing eerst uit geprogrammeerd zou moeten worden.

Om wel overbodige vragen over te kunnen slaan en in zo weinig mogelijk stappen tot een artikel te komen moest ik iets anders bedenken. Hiervoor heb ik een brainstorm sessie gehad met mijn bedrijfsmentor om ideeën op te doen. Uit deze brainstorm sessie is bij mij het idee ontstaan om een groepen hiërarchie te creëren. Het idee hierachter zal ik eerst beschrijven en vervolgens zal ik laten zien hoe dit in het klassendiagram is verwerkt.

Vanuit het startpunt “Oplossing” zal ik m.b.v. illustraties stap voor stap laten zien hoe er effectief tot een artikel kan worden gekomen.

Stap 1

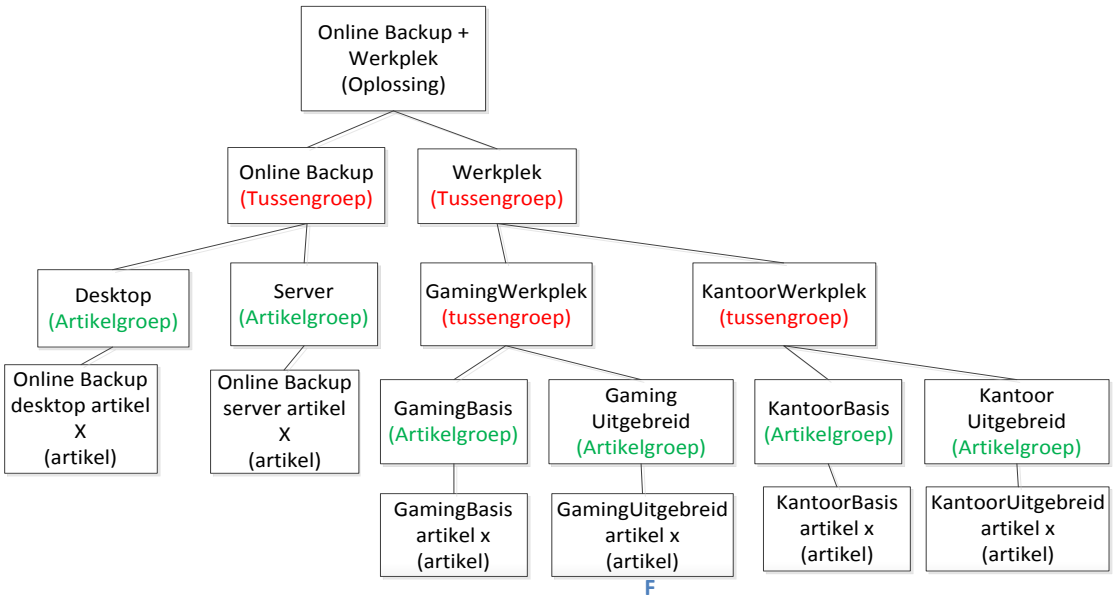
De eerste stap is dat de oplossing gesplitst wordt in groepen, welke tussengroepen worden genoemd. De eerste splitsing vanaf oplossing moet het grootste verschil zijn in de oplossing. Kijkend naar het voorbeeld in figuur 14 is het grootste verschil in de oplossing “Online Backup & Werkplek “: Online Backup en Werkplek. Dit worden dan de eerste 2 tussengroepen.



Figuur 14: Stap 1 in groepen hiërarchie

Stap 2

Als er nog maar 1 essentieel verschil is vanaf een tussengroep tot een artikel dan is de volgende groep een artikelgroep. Is er nog wel meer dan 1 groot verschil dan is de volgende groep na een tussengroep een volgende tussengroep. In figuur 15 is dit geïllustreerd. Na de tussengroep “Online Backup” is er nog maar 1 essentieel verschil: Desktop of Server, dit worden dan 2 artikelgroepen(groen weergegeven). Vanaf tussengroep “werkplek” zijn er nog wel grotere verschillen namelijk: Kantoorwerkplek of Gaming werkplek, dit worden dan 2 tussengroepen(rood weergegeven). Na tussengroep kantoorwerkplek en gaming werkplek zijn er geen grote verschillen meer dus daaronder komt een artikelgroep.



F

Figuur 15: Stap 2 In groepen hiërarchie

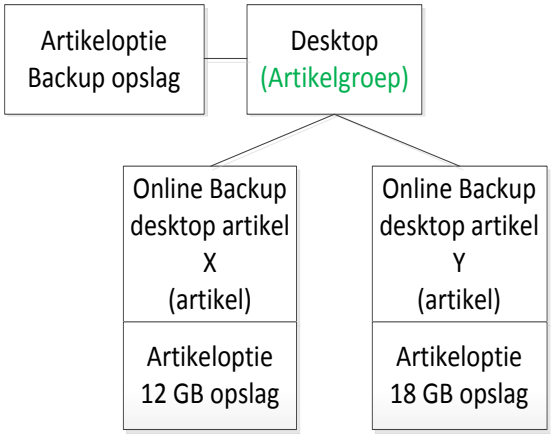
Stap 3

In figuur 15 is onder de artikelgroep telkens maar 1 artikel te zien maar in werkelijkheid zullen dit meerdere artikelen zijn. KBITE levert immers meer dan bijvoorbeeld 1 online backup artikel. Om dit te verduidelijken licht ik het stuk vanaf artikelgroep “Desktop” uit in figuur 16, als voorbeeld worden 2 fictieve artikelen(X en Y) gebruikt.

Alle artikelen hebben specifieke kenmerken die een artikel uniek maken. Deze kenmerken worden artikelopties genoemd. Hierbij moet worden gedacht aan:

- Backup opslagcapaciteit(bv. 12gb, 24gb, 50gb)
- Resolutie(bv. 1200x1024, 800x600)
- Kleur(bv. Geel, Blauw, Rood)

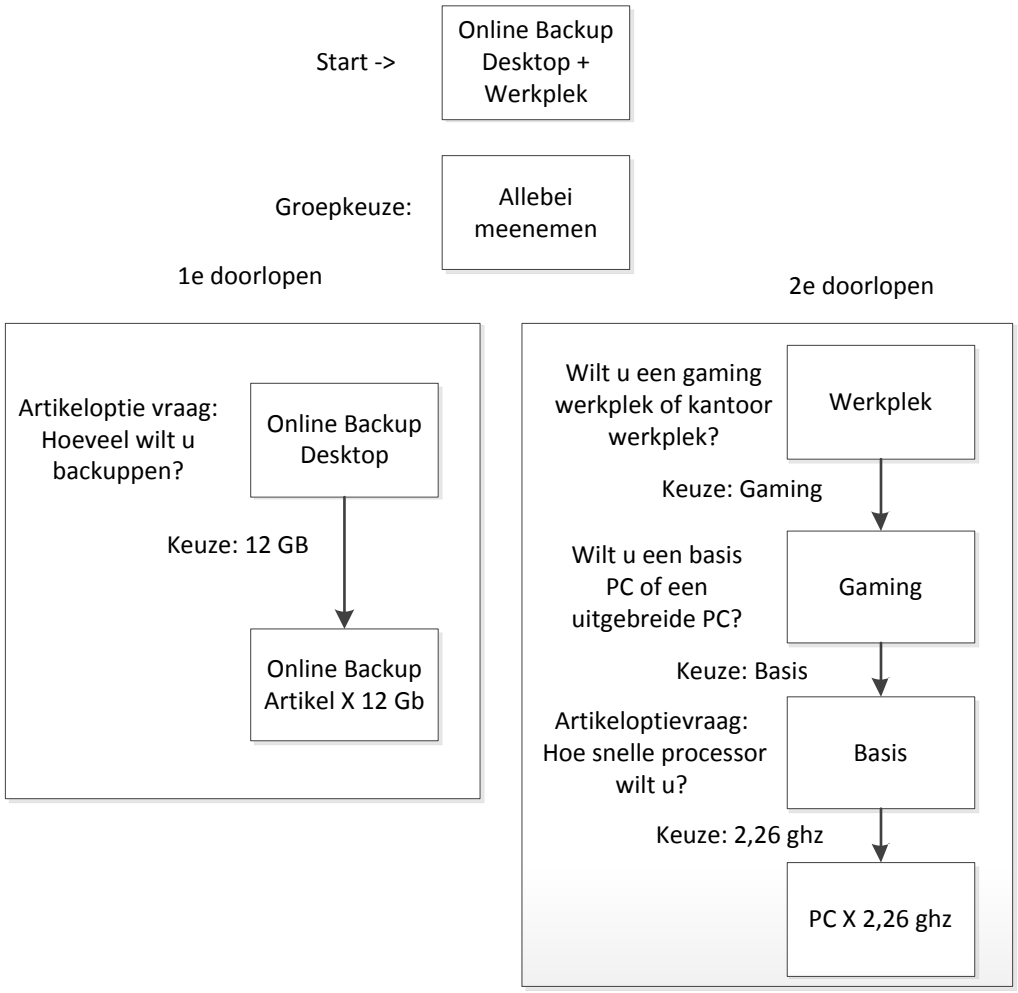
In het voorbeeld van figuur 16 zitten er 2 artikelen in de artikelgroep desktop. Om hieruit een keuze te kunnen maken zal er gevraagd worden naar de artikeloptie Backup Opslag. De gebruiker heeft dan de mogelijkheid om tussen 12GB opslag of 18GB opslag te kiezen.



Figuur 16: Stap 3 in groepen hiërarchie

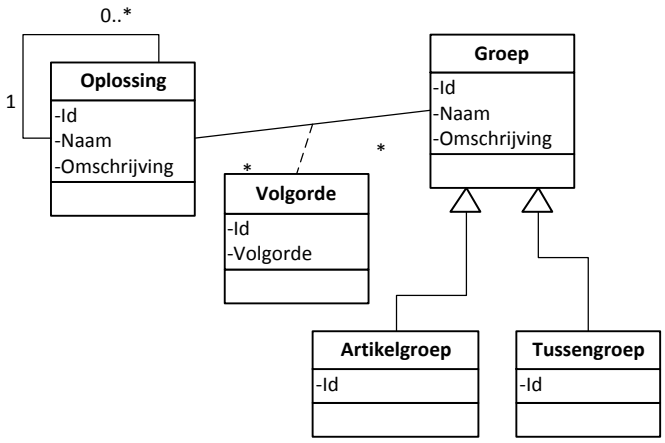
In de praktijk

In de genoemde voorbeelden heb ik steeds de oplossing Online Backup + Werkplek genoemd om duidelijk de verschillen in de hiërarchie aan te geven. In de praktijk zou het echter nog effectiever zijn om de oplossing Online Backup Desktop + werkplek te configureren. De artikelgroep desktop zou dan direct onder de oplossing vallen. Hierdoor wordt er nog beter omgegaan met overbodige vragen want er hoeft niet worden gevraagd naar desktop of server, wat overbodig zou zijn want een desktop is een werkplek. In de praktijk is elke groep een antwoord op een wizard vraag, totdat een keuze een artikelgroep is dan worden er artikeloptie vragen gesteld om de keuze voor een artikel te specificeren. In figuur 17 is dit geïllustreerd.



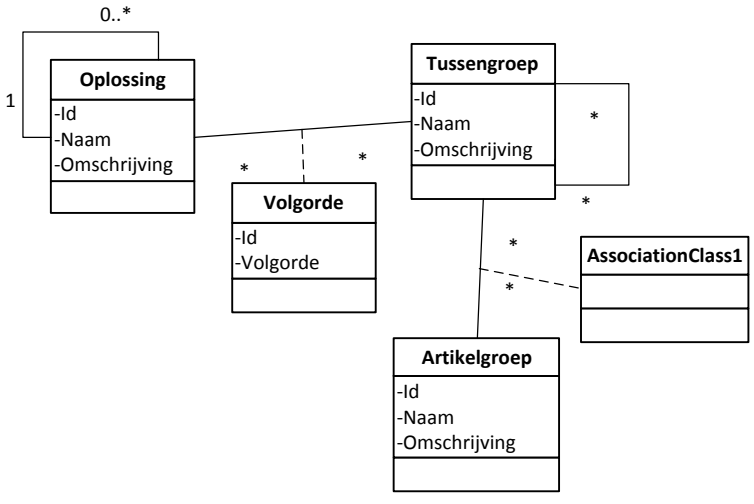
Figuur17: Illustratie gebruik van groepen in wizard

Om het gehele groepen gedeelte naar het klassendiagram te vertalen zijn er meerdere versies opgesteld. In een eerste versie heb ik een groep vertaald naar een specialisatie van artikelgroep en tussengroep. Op dat moment wist ik dat een artikelgroep en een tussengroep een aantal gemeenschappelijke attributen hadden en dat een attribuut “vraag” waarschijnlijk een van de type groepen specifiek zou maken om artikelen te kunnen laten afvallen. Deze eerste concept versie is weergegeven in figuur 18.



Figuur 118: 1e Concept versie

Dit vraagstuk heb ik besproken met mijn bedrijfsmentor omdat ik er niet zeker van was dat dit de juiste manier zou zijn. Met name over het stuk dat een artikelgroep en tussengroep allebei onder een oplossing kunnen vallen en het niveau verschil van Oplossing -> tussengroep -> artikelgroep weg lijkt te zijn. Om hierover meer duidelijk te krijgen heb ik nog een versie gemaakt met een andere constructie waarbij dit niveau verschil duidelijker zichtbaar was. In deze versie, weergegeven in figuur 19, heb ik een tussengroep en artikelgroep los van elkaar gemodelleerd waarbij tussengroepen kunnen worden herhaald door een parent-child relatie. Artikelgroepen konden naar mijn inziens onder meerdere tussengroepen vallen en een tussengroep kan meerdere artikelgroepen bevatten. Tevens kan een tussengroep onder meerdere andere tussengroepen vallen.



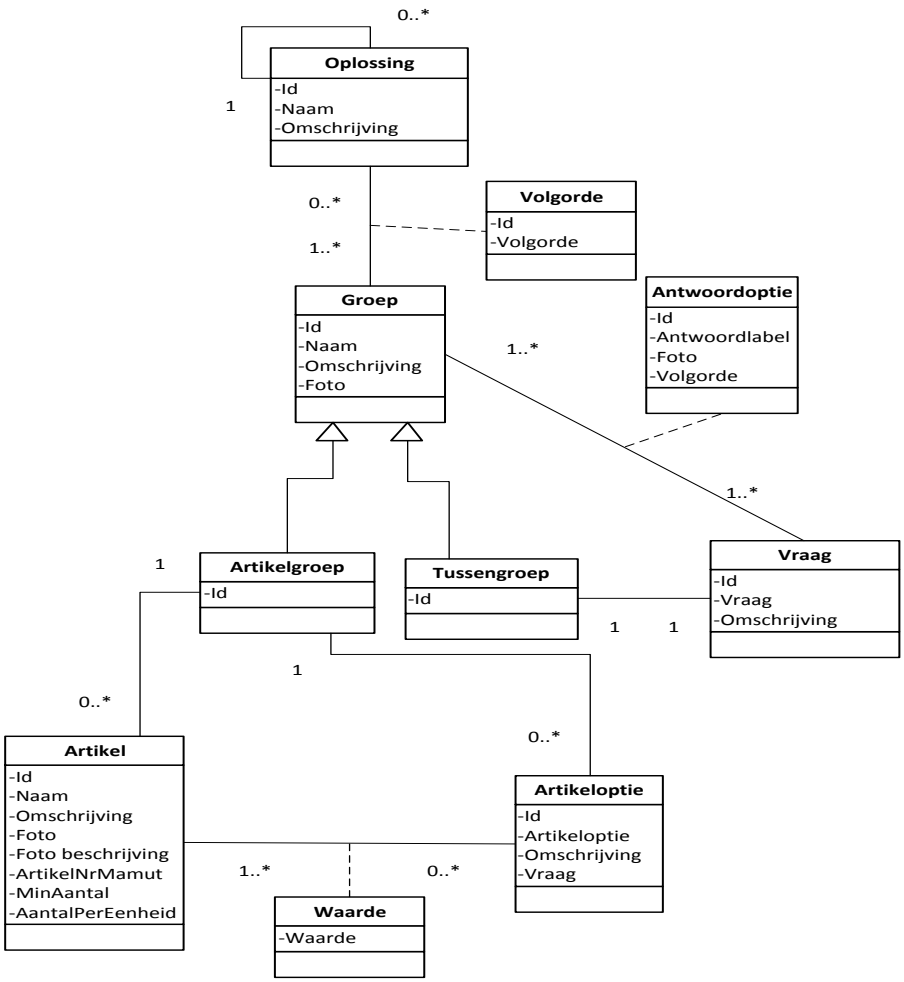
Figuur 1912: 2e concept versie

Hier zag ik echter ook een aantal obstakels in terug komen. Het zou namelijk beteken er een veel op veel relatie tussen de parent en child zou komen, wat vrij ongebruikelijk is en veel joins tussen tabellen zou opleveren dat de performance niet ten goede zou komen. Omdat dit een nogal belangrijk stuk was van mijn project heb ik hiervoor gesprek gepland met mijn opdrachtgever en bedrijfsmentor. Mijn insteek voor dit gesprek was om het eens te worden over de modellering van de groepen en of de opdrachtgever vanuit functioneel oogpunt hiermee zou kunnen werken.

In dit gesprek is de indeling van tussengroepen, artikelgroepen tot een artikel voor het product IP-Camerabeveiliging volledig uitgewerkt. Dit was ook een product waar de opdrachtgever incorrecte en in een onlogische volgorde vragen bij dacht om tot een artikel uit te komen. Bijvoorbeeld "Wilt u HD opnemen?" en de gebruiker kiest voor nee, dan zouden er op het einde toch HD camera's moeten kunnen worden getoond als de klant hier interesse voor heb. Ik heb de opdrachtgever duidelijk gemaakt dat er hierdoor een vicieuze cirkel ontstaat en er nooit bij een artikel kan worden uitgekomen. Aan de hand van een artikellijst heb ik de opdrachtgever laten zien dat er keuzes moeten worden gemaakt om tot een artikel uit te komen. En dat het gebruik van groepen hier ideaal voor is omdat deze ook hergebruikt kunnen worden in verschillende offerte oplossingen. Het was leuk om te zien dat de opdrachtgever hierdoor enthousiaster werd en de voordelen van het gebruik van groepen ging inzien. Tevens zorgde dit ook bij mij voor nieuwe inzichten om het technische gedeelte in te vullen. Het bleek namelijk dat de tweede versie niet geschikt zou zijn omdat een artikelgroep wel direct onder een oplossing moet kunnen vallen. Als artikelen binnen een artikelgroep namelijk geen grote verschillen hebben zal deze groep direct onder een oplossing moeten komen.

Om de wizard vragen te kunnen plaatsen in het model stelde ik voor om een aparte klasse vraag hiervoor te modellen en de specialisatie en artikelgroep en tussengroep uit versie 1 aan te houden. Mijn bedrijfsmentor gaf mij aan dat ik er wel rekening mee moest houden dat deze groepen geen specifieke attributen zouden hebben. Dit heb ik toen meegenomen in mijn overweging, echter heb ik ervoor gekozen om toch deze specialisatie aan te houden om hun verschillende relaties met andere klasses duidelijk te maken.

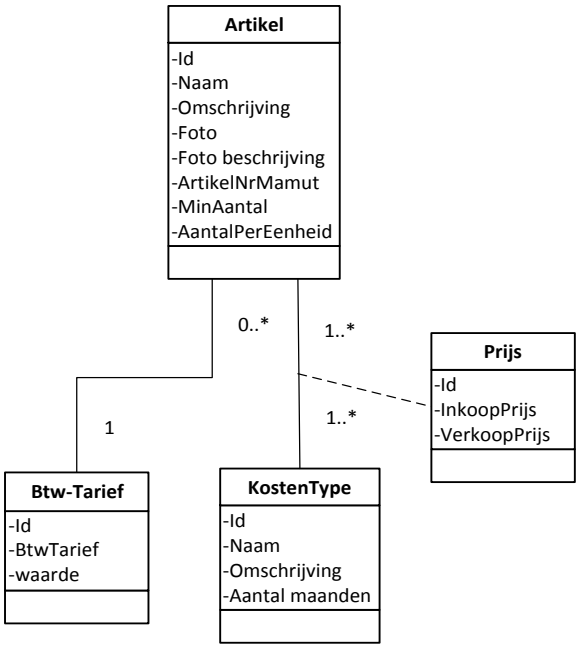
Na het uitwerken van dit gedeelte van het klassendiagram is het geverifieerd door mijn bedrijfsmentor en opdrachtgever, waarbij iedereen het eens was dat dit de geschikte modellering en praktische invulling was om mee verder te gaan. Ter verduidelijking is de definitieve versie van dit gedeelte van het klassendiagram weergegeven in figuur 20 op de volgende bladzijde.



Figuur 20 Definitieve versie Oplossing tot Artikel

Om het beheer van de applicatie gemakkelijk te houden heeft artikelgroep ook een relatie met artikeloptie zodat bij het toevoegen van een nieuw artikel alleen de waarde van de artikeloptie moet worden ingevuld. Tevens is in de assosiatieklasse antwoordoptie het attribuut antwoordlabel en foto opgenomen. Dit is ervoor om de gebruiker tijdens de wizard vragen te ondersteunen. Foto's kunnen namelijk een antwoordmogelijkheid verduidelijken en antwoordlabel is ervoor dat de gebruiker geen onduidelijke groepsnamen krijgt te zien.

De opdrachtgever had ook eisen dat arbeidskosten voor installatie/configuratie van een artikel moeten kunnen worden toegevoegd en dat een artikel meerdere kostenopties kan bevatten zoals eenmalige betaling of maandelijkse betaling. Dit heb ik opgelost door een aparte klasse kostentype op te nemen. Een artikel kan zodoende meerdere kostentypes bevatten en een kostentype kan bij meerdere artikelen horen. Dit zorgt voor uitstekende uitbreidbaarheid van de offertetool op kostengebied. In figuur 21 is dit gedeelte van het klassendiagram weergegeven.

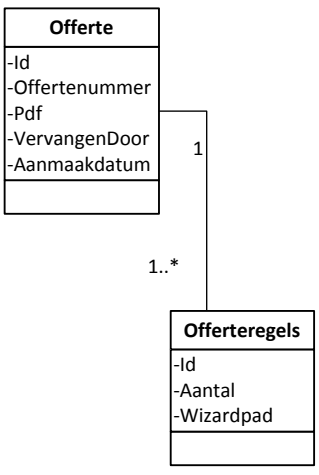


Figuur 21 Artikel met kostentypes

Met betrekking tot het wijzigen van offertes had de opdrachtgever o.a. de volgende eisen:

- Het systeem moet zo veel mogelijk ingevulde gegevens onthouden bij het wijzigen tijdens het wizard proces
- Een gewijzigde offerte kan alleen worden opgeslagen naar een nieuwe versie

In het klassendiagram heb ik dit opgelost door een attribuut PDF op te nemen. Een offerte wordt opgeslagen in PDF en kan dus niet meer worden gewijzigd. Tevens wordt bij het opslaan van een offerte de ID's van de doorgelopen groep opgeslagen in het attribuut "wizardpad".Op deze manier moet een offerte in een nieuwe versie worden opgeslagen. En tijdens het wijzigen worden de keuzes van te voren ingevuld aan de hand van het opgeslagen wizard pad.
Ter verduidelijking is dit gedeelte van het klassendiagram weergegeven in figuur 22.



Figuur 22 Gedeelte klassendiagram m.b.t. offertes

Het opstellen van het klassendiagram heb ik in eerste instantie gedaan met Oracle Jdeveloper. Hiermee is het mogelijk om een UML klassendiagram op te stellen en deze vervolgens om te zetten naar een implementatiemodel. Dit implementatiemodel kan vervolgens direct worden uitgevoerd om de database te implementeren, wat veel tijd kan besparen. Echter op een geven moment wilde ik een oudere versie van het klassendiagram terug kijken maar deze bleek toen onleesbaar/overschreden wegens een reeds onbekende reden. Doordat ik gelukkig alle versies van het klassendiagram op papier had was de informatie niet verloren gegaan.

Ik heb toen wat tijd gestoken om de oorzaak hiervan te achterhalen en het probleem op te lossen maar dit was niet gelukt. Dit betekende dat ik 2 keuzes had: Meer tijd besteden om te onderzoeken hoe versies kunnen worden beheerd met Jdeveloper, in de wetenschap dat ik het klassendiagram makkelijk over kan zetten naar een implementatiemodel. Of het klassendiagram maken met Visio waarvan ik zeker wist hoe verschillende versies kunnen worden beheerd maar als nadeel dat ik het handmatig moet overzetten naar een implementatiemodel in Jdeveloper.

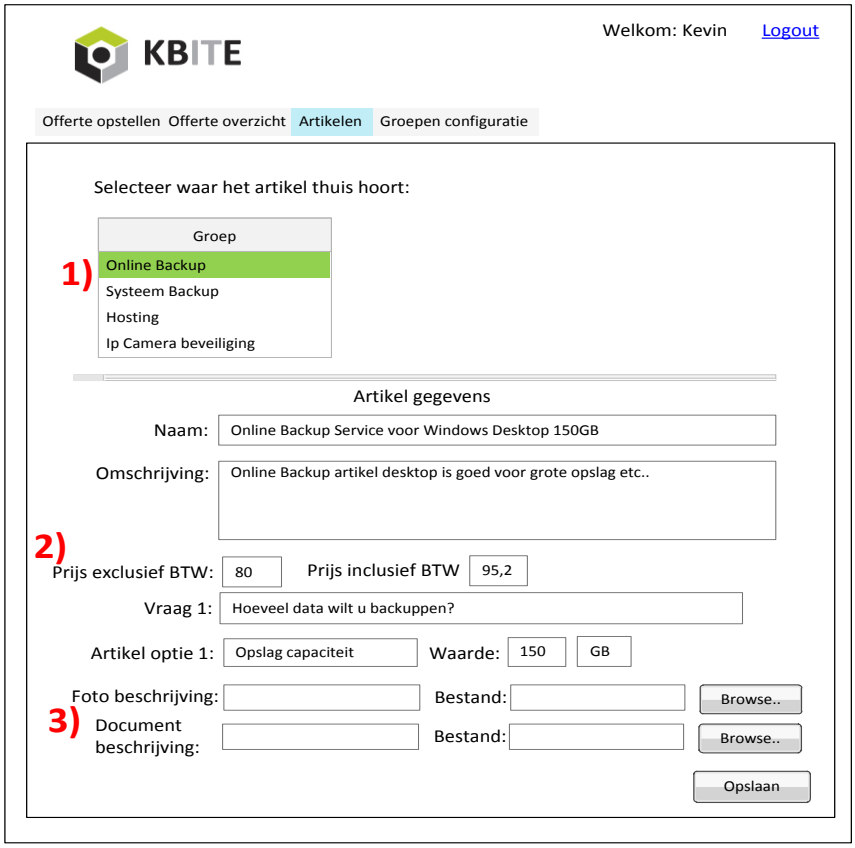
Ik vond het niet waard om nog meer tijd te steken in het onderzoeken van versiebeheer in Jdeveloper en heb er daarom voor gekozen om verder te gaan met Visio. Dit had naar mijn inziens ook wel een voordeel. Door het handmatig overzetten kan direct een dubbele controle worden uitgevoerd, enerzijds of er geen fouten meer zitten in het klassendiagram en anderzijds of het implementatiemodel volledig is. Het modelleren van het groepen gedeelte was een behoorlijk lastig stuk en hiermee zijn ook verschillende versies gemoeid. Het was daarom belangrijk dat ik de verschillende versies op orde had zodat ik hierop terug kan vallen als een oudere versie toch beter blijkt te werken.

8.3 Functioneel ontwerp applicatie maken

Een van de activiteiten die ik heb uitgevoerd is het maken van een functioneel ontwerp van de applicatie. Voor het maken van een functioneel ontwerp heb ik gebruik gemaakt van wireframes, deze heb ik gemaakt met behulp van Visio. Een wireframe geeft visueel weer welke gegevens op welke pagina's in de applicatie moeten komen. Tevens zorgt het voor duidelijkere communicatie met de opdrachtgever. Omdat het visueel weergeven van complexe zaken vaak meer duidelijkheid geeft heb ik in een zo vroeg mogelijk stadium het functioneel ontwerp besproken. Het belangrijkste aan het functioneel ontwerp was dat deze weergeeft hoe het aanvragen van offertes verloopt en hoe het configureren van dit proces verloopt.

In figuur 23 is de eerste versie van het functioneel ontwerp te zien m.b.t. een artikel toevoegen. Deze versie is opgezet aan de hand van de eisen die zijn opgesteld in de inception fase en naar aanleiding van de groepen hiërarchie die beschreven is in hoofdstuk 8.2 .Ik zal hier de 3 interessantste punten van deze eerste versie toelichten:

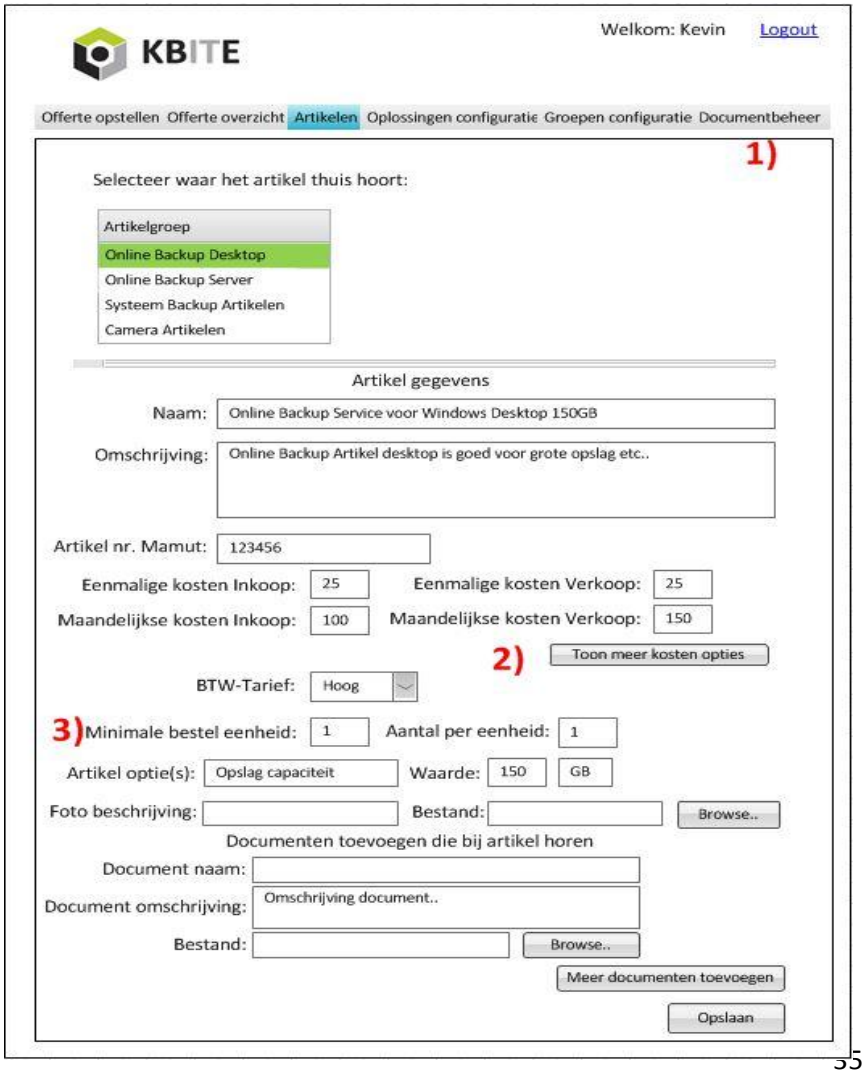
1) Bij het toevoegen van een artikel is het niet alleen een kwestie van artikelgegevens invoeren maar zal er ook moeten worden aangegeven onder welke groep het artikel thuis hoort. Door het selecteren van de groep wordt automatisch bijbehorende artikeloptie vraag en artikeloptie getoond(deze worden namelijk bij het invoeren van een artikelgroep ingevoerd en zijn dus bekend) en hoeft alleen de waarde te worden ingevuld.



Figuur 23 Eerste versie Functioneel ontwerp artikel toevoegen

2) Bij het bespreken van deze versie leverde punt 2 een interessante gedachte op bij de opdrachtgever. In deze versie is te zien dat een prijs inclusief en exclusief BTW kan worden ingevoerd. Echter in het gesprek met mij gaf hij aan dat dit niet voldoende was omdat er bij een hoop aantal artikelen verschillende kosten opties zijn zoals: eenmalige, maandelijkse, per kwartaal en ook een verschil btw tarief. Dit waren een aantal verbeter punten voor mij om te verwerken in een volgende versie.
3) Een belangrijke eis van de opdrachtgever was dat er een document met bijvoorbeeld toelichting over specificaties van het artikel kan worden toegevoegd. Bij punt 3 is te zien dat deze kan worden toegevoegd inclusief een document beschrijving.
Het bespreken van deze eerste versie leverde ook een discussiepunt op over het meesturen van algemene voorwaarden,wat ook een vereiste is. Algemene voorwaarden bestaan uit modules bijvoorbeeld software module, of hardware module. Elke module is een apart PDF bestand. De opdrachtgever wilde deze modules apart bij een artikel toevoegen, zodoende dat ze mee worden gestuurd bij een artikel. Naar mijn inziens kon dit beter worden opgelost door alle modules(PDF bestanden) samen te voegen tot 1 algemene voorwaarden bestand(1 PDF bestand) zodat deze maar 1x hoeft worden toegevoegd en altijd met een offerte wordt meegestuurd.

In figuur 24 is de definitieve versie van het functioneel ontwerp te zien m.b.t. artikel toevoegen. De belangrijkste verbeteringen ten opzichte van de eerste versie zal ik hier toelichten.



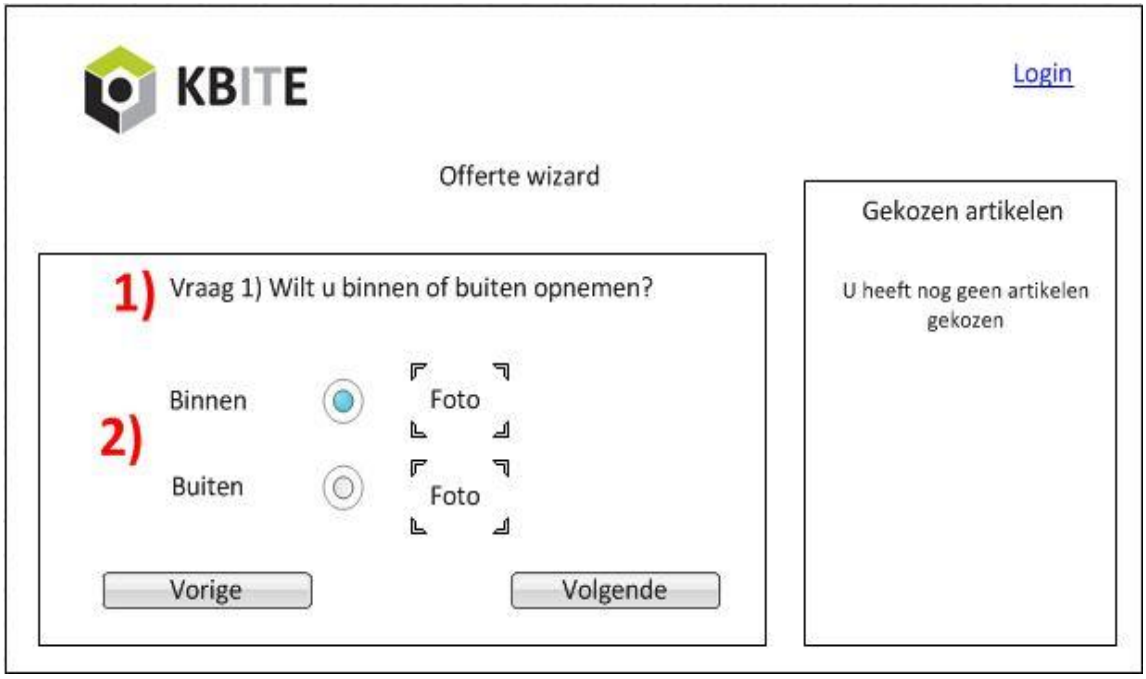
Figuur 24 Definitieve versie Functioneel ontwerp

1) Bij punt 1 is in de definitieve versie een extra tabblad te zien: documentbeheer . Documenten die niet specifiek bij een artikel horen kunnen op die pagina worden ingevoerd. Het gaat dan voornamelijk om de eerder genoemde algemene voorwaarden. Deze hoeven dan maar 1x te worden ingevoerd en niet elke keer als een nieuw artikel wordt ingevoerd en als de algemene voorwaarden wijzigen hoeft ook maar 1x het nieuwe document worden ingevoerd.
2) Bij punt 2 is de verbetering te zien m.b.t. de eerder genoemde kostenopties. De eenmalige en maandelijkse kosten worden standaard getoond bij het invoeren van een artikel. Als men op de knop “Toon meer kosten opties” klikt dan worden extra kosten opties getoond die kunnen worden ingevuld zoals: kwartaal kosten of jaarlijkse kosten. Tevens heb ik hier gelijk een andere eis van de opdrachtgever verwerkt namelijk: “Het toevoegen van arbeidskosten voor installatie/configuratie van artikelen”. Men krijgt namelijk ook de mogelijk om deze in te voeren als op de knop “Toon meer kostenopties” wordt geklikt.
3) Bij punt 3 staan ook 2 nieuwe invoervelden ten opzichte van de eerste versie, namelijk: minimale bestel eenheid en aantal per eenheid. Deze zijn ervoor om te kunnen voldoen aan alle artikelen die kunnen worden ingevoerd. Bij licenties is het regelmatig het geval dat er een minimale afname is of alleen per 5 bijvoorbeeld worden verkocht.

Hieronder zal ik nog 2 schermen uit het functioneel ontwerp m.b.t. het opstellen van offertes toelichten.

Figuur 25 Definitieve versie functioneel ontwerp offerte aanvragen

1) Bij punt 1 is te zien dat zodra een offerte oplossing wordt gestart de eerste groepen worden getoond die meegenomen kunnen worden in de offerte wizard. Zoals uitgelegd in hoofdstuk 8.2.
2) Bij punt 2 is de visuele weergave te zien van een winkelwagen. Met zo’n winkelwagen wordt voldaan aan de eis dat er meerdere artikelen aan een offerte kunnen worden toegevoegd.



Figuur 26 Definitieve versie functioneel ontwerp offerte aanvragen

1) In figuur 25 is te zien dat de groep “IP Camera’s” is geselecteerd als er dan op volgende wordt geklikt krijgt men het scherm van figuur 26 te zien. Bij punt 1 wordt de vraag getoond die hoort bij die groep.
2) Bij punt 2 zijn de groepen te zien die in de uitgelegde groepen hiërarchie onder de groep “IP Camera’s” vallen. Onder de groep “IP Camera’s” vallen dus 2 groepen : Binnen en Buiten. Men moet dus een keuze tussen 1 van de twee maken en vervolgens op “Volgende” klikken. In figuur 26 is de groep “Binnen” geselecteerd en als op “Volgende” wordt geklikt wordt hetzelfde principe herhaald: De vraag die hoort bij de groep Binnen wordt getoond en de onderliggende groepen waar weer een keuze tussen kan worden gemaakt.

8.4 Relationeel Representatiemodel opstellen

In het relationeel representatiemodel worden de toekomstige tabellen met de desbetreffende attributen weergegeven. Voor het omzetten van de klasses groep, artikelgroep en tussengroep had ik 3 mogelijkheden:

- Voor elke klasse 1 tabel
- Alle klasses in 1 tabel
- 1 tabel per subklasse(1 tabel voor artikelgroep, 1 tabel voor tussengroep)

Om hierin een juiste keuze te maken heb ik de voor en nadelen van de 3 mogelijkheden op een rij gezet en aan elkaar afgewogen om een beslissing te maken. Deze voor en nadelen zijn terug te vinden in onderstaande tabel.

	Voordeel	Nadeel
Alles in 1 tabel	Geen redundante attribuut definities	Ruimte voor niet ingevulde attribuut definities wordt verspeeld
	Geen zoekproces nodig voor specifieke groep te reconstrueren (geen join)	
		Geen overervingstructuur meer te vinden
Voor elke klasse 1 tabel	Geen redundante attribuut definities	Zoek proces nodig om specifieke groep te reconstrueren(Join nodig)
	Volgt overervingstructuur	
1 tabel per subklasse (1 tabel voor artikelgroep en 1 tabel voor tussengroep)	Geen zoekproces nodig voor specifieke groep te reconstrueren (geen join)	
		Redundante attribuut definities
		Geen overerving structuur terug te vinden
		Om generieke type groep terug te vinden moeten meerdere tabellen worden doorzocht

Figuur 27 Voor en Nadelen omzetten van groepen

De voordelen van de tweede mogelijkheid: voor elke klasse 1 tabel was voor mij geen optie. Dat dit geen redundante attribuut definities zou opleveren is voor dit project te verwaarlozen. Tevens zou het volgen van een overervingstructuur ook geen voordeel voor dit project opleveren omdat er alleen gemeenschappelijke attributen zijn en alleen de relaties verschillen. De derde mogelijkheid: 1 tabel per subklasse vond ik ook geen geschikte mogelijkheid. Het voordeel hiervan was wel dat er geen join nodig is om een specifieke groep te reconstrueren. Echter zou er wel een join op meerdere tabellen nodig zijn om de volgende groep te kunnen bepalen, wat bij

de eerste mogelijkheid niet het geval is. Voor mij was de eerste mogelijkheid de beste keuze, dit neemt niet alleen bovengenoemde voordelen met zich mee. Maar betekent ook dat er maar 1 tabel nodig is en met Oracle Apex is hierop redelijk eenvoudig een formulier en een overzichts pagina op te maken. Om ook te denken aan de uitbreidbaarheid en onderhoudbaarheid van de applicatie is ook groep type als aparte tabel opgenomen. Mochten er in de toekomst andere typen groepen bij komen dan kunnen deze gemakkelijk worden uitgebreid. In het klassendiagram was dit al gedaan met bijvoorbeeld document type en eenheid. In figuur 28 is een gedeelte van het Relationeel Representatie model weergegeven, voor het hele model verwijs ik naar de bijlage.

Groep(id,naam,omschrijving,*groep_type*)
groep_type is vreemde sleutel en verwijst naar id de tabel GroepType, null niet toegestaan

GroepType(id,naam)

Artikel(id,naam,omschrijving,FotoLink,FotoOmschrijving,ArtikelNrMamut,MinAantal,AantalPerEenhe
id,*btwTarief_id*,*Groep_id*)

btwTarief_id is vreemde sleutel en verwijst naar id in de tabel BtwTarief, null is niet toegestaan
groep_id is vreemde sleutel en verwijst naar id in de tabel Groep, null is niet toegestaan

Artikeloptie(id,artikeloptie,omschrijving,vraag,*eenheid_id*,*groep_id*)

Waarde(*artikeloptie_id*,*artikel_id*,waarde)

artikeloptie_id is vreemde sleutel en verwijst naar id in de tabel Artikeloptie, null is niet toegestaan
artikel_id is vreemde sleutel en verwijst naar id in de tabel Artikel, null is niet toegestaan

BtwTarief(id,BtwTarief,waarde)

Figuur 28 Gedeelte Relationeel Representatiemodel

8.5 Relationeel Implementatiemodel

Het relationeel implementatiemodel geeft precies weer hoe de tabellen van de database eruit komen te zien. Voor het opstellen van het implementatiemodel heb ik Jdeveloper gebruikt. Met Jdeveloper is het mogelijk om in een visuele weergave de tabellen en relaties aan te maken en vervolgens hiervan een script te laten genereren die speciaal geschikt is voor de Oracle database 11 G. Het gebruik van Jdeveloper bespaart niet alleen tijd maar zorgt ook voor minder foutgevoeligheid doordat het script automatisch wordt gegenereerd.

Ik heb dit ook toegepast voor dit project. Aan de hand van het relationeel representatiemodel en klassendiagram heb ik alle tabellen en hun relaties aangemaakt. Vervolgens heb ik van de visuele weergave hiervan een script gegenereerd die direct implementeerbaar is. Dit script heb ik ook bewaard zodat in de toekomst bij een calamiteit deze direct gebruikt kan worden om de database te implementeren. Een gedeelte van het implementatiemodel is te zien in figuur 29, voor het gehele implementatiemodel verwijs ik naar de bijlage.

Voor het opslaan van tekst heb ik gekozen voor het datatype Varchar2.Het grote voordeel van Varchar2 is dat het geen ruimte in beslag neemt voor NULL waardes. Dit in tegenstelling tot Varchar die wel ruimte in beslag neemt voor niet gebruikte waardes. Voor de opslag van prijzen en andere numerieke waardes heb ik gekozen het datatype Number zodat hiermee gerekend kan worden. Het datatype BLOB wordt gebruikt voor de opslag van foto's en bestanden. BlOB kan grote databestanden opslaan in de database als binaire data.

```
CREATE TABLE ARTIKEL
(
 ID NUMBER NOT NULL
,NAAM VARCHAR2(200) NOT NULL
,OMSCHRIJVING  VARCHAR2(200) NOT NULL
,FOTO VARCHAR2(200)
,FOTOBESCHRIJVING  VARCHAR2(200)
,ARTIKELNRMAMUT  NUMBER
,MINAANTAL  NUMBER NOT NULL
,AANTALPEREENHEID  NUMBER NOT NULL
,BTWTARIEF_ID  NUMBER NOT NULL
,GROEP_ID NUMBER NOT NULL
,CONSTRAINT ARTIKEL_PK PRIMARY KEY
 (
  ID
 )
 ENABLE
);

CREATE TABLE ARTIKELOPTIE
(
 ID NUMBER NOT NULL
,ARTIKELOPTIE VARCHAR2(200) NOT NULL
,OMSCHRIJVING  VARCHAR2(200) NOT NULL
,VRAAG VARCHAR2(200) NOT NULL
,EENHEID_ID  NUMBER NOT NULL
,CONSTRAINT ARTIKELOPTIE_PK PRIMARY KEY
 (
  ID
 )
 ENABLE
);
```

Figuur 29: Gedeelte Implementatiemodel

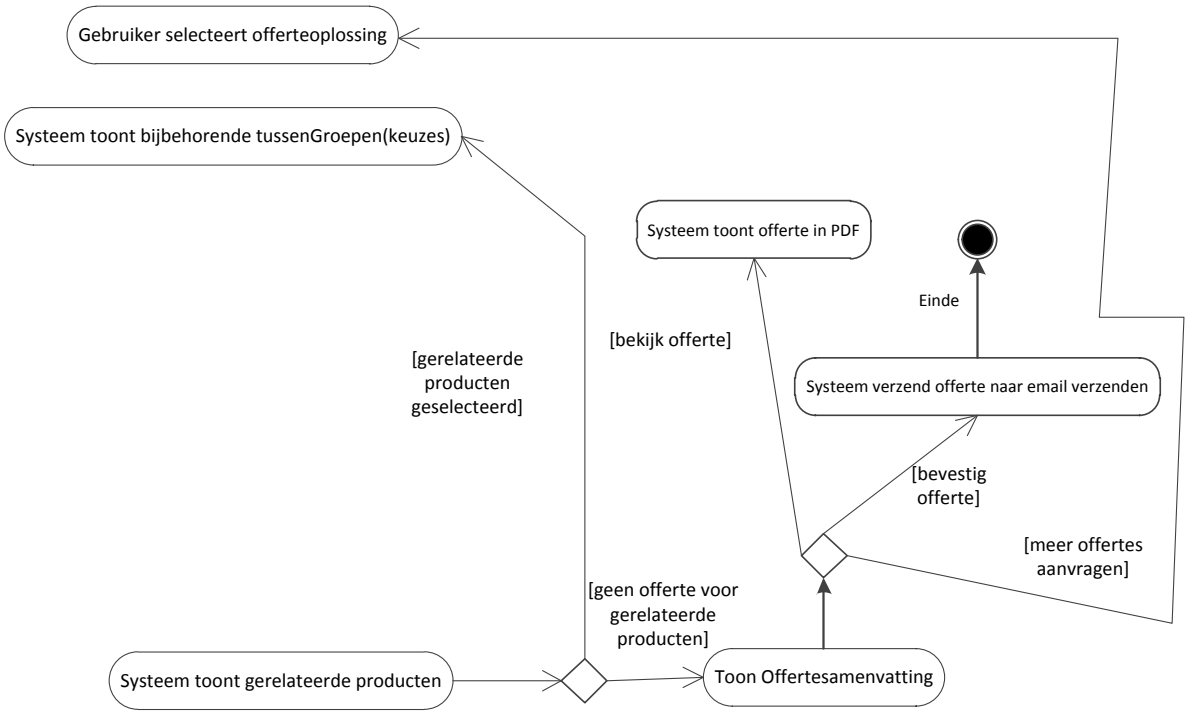
8.6 Activitydiagram

Om het gedrag van de applicatie weer te geven heb ik een activitydiagram opgesteld. Hierin heb ik alle acties beschreven die door het systeem worden uitgevoerd als men de offertewizard start. Dit diagram is niet direct aan de opdrachtgever gericht maar is gebruikt om mijzelf te helpen bij het realiseren van de offerte wizard.

Tijdens dit proces moet het systeem een aantal acties uitvoeren die in de applicatie moeten worden gerealiseerd. Door dit visueel weer te geven in een activitydiagram blijft de volgorde van deze acties overzichtelijk en kunnen pagina's in de applicatie hierop worden afgestemd.

In figuur 30 is een gedeelte van dit diagram weergegeven. Hierin komen een aantal eisen van de opdrachtgever in terug namelijk:

- Het systeem moet een offerte naar email verzenden
- Gerelateerde producten kunnen selecteren
- Het aanvragen van meerdere offertes in 1 keer



Figuur 30: Gedeelte Activitydiagram

8.7 Database implementatie

Om het implementatiemodel te kunnen implementeren heb ik eerst een nieuwe user(schema) aangemaakt genaamd “offertetool” en vervolgens heb ik deze user connectie rechten gegeven. Hiermee kon ik met Jdeveloper connectie maken naar de database en het script uitvoeren. Na het uitvoeren van dit script moest ik nog enkele aanpassingen doem:

Toevoegdatum & wijzigdatum

Mijn bedrijfsmentor gaf bij het zien van de database het advies om nog een toevoegdatum en wijzigdatum bij elke tabel aan te maken, zodat er bijgehouden kan worden wanneer er mutaties op de database zijn verricht. Tevens is dit handig te gebruiken voor het testen van de functionaliteiten omdat er zo zichtbaar wordt of een actie daadwerkelijk plaats heeft gevonden.

Om de toevoegdatum en wijzigdatum automatisch te laten invullen heb ik twee triggers aangemaakt. Één trigger gaat af bij het toevoegen van een record en vult daarbij de toevoegdatum met de systeem datum. De andere trigger gat af bij het wijzigen van een record en vult op dat moment de wijzigdatum met de systeem datum. Een voorbeeld van een trigger is weergegeven in figuur 31.

```
create or replace trigger update_toevoegdatum_artikel
 before insert on artikel
 for each row

declare
begin

select seq_auto_inc_artikel.nextval into :new.id from dual;

:new.toevoegdatum := sysdate;

end update_toevoegdatum_artikel;
```

Figuur 31: Trigger toevoegdatum artikel tabel

Auto increment

Met andere databases waarmee ik heb gewerkt was ik gewend om een auto increment veld te kunnen aangeven voor de primary keys. Na wat onderzoek op internet en navraag bij mijn bedrijfsmentor bleek dit niet zo te zijn. Ik heb dit opgelost door voor elke tabel, die dit nodig had, een sequence aan te maken. Een sequence genereert unieke getallen die voor primary keys gebruikt kunnen worden. Een voorbeeld hiervan is weergegeven in figuur 32. De trigger die afgaat bij het toevoegen van een record voert vervolgens een functie uit, die een nieuwe getal aan de sequence vraagt. Voor de tabel artikel ziet deze functie er als volgt uit:
select seq_auto_inc_artikel.nextval into :new.id from dual;

```
create sequence SEQ_AUTO_INC_ARTIKEL
minvalue 1
maxvalue 9999999999999999999999999999
start with 1
increment by 1
nocache;
```

Figuur 32: Sequence artikel tabel

9. Construction fase

In de construction fase is het implementatiemodel geïmplementeerd en is er gewerkt aan het bouwen van de applicatie. In dit hoofdstuk zal ik een beschrijving geven over de implementatie van de database en het bouwen van de applicatie. Voordat ik zal beschrijven hoe de diverse functionaliteiten zijn gebouwd zal ik eerst nog enkele onderdelen van Oracle Apex uitlichten.

9.1 Database implementatie

Om het implementatiemodel te kunnen implementeren heb ik eerst een nieuwe user(schema) aangemaakt genaamd “offertetool” en vervolgens heb ik deze user connectie rechten gegeven. Hiermee kon ik met Jdeveloper connectie maken naar de database en het script uitvoeren.Na het uitvoeren van het script moest ik nog wat kleine aanpassingen doen:

Toevoegdatum & wijzigdatum

Mijn bedrijfsmentor gaf bij het zien van de database het advies om nog een toevoegdatum en wijzigdatum bij elke tabel aan te maken, zodat er bijgehouden kan worden wanneer er mutaties op de database zijn verricht. Tevens is dit handig te gebruiken voor het testen van de functionaliteiten omdat er zo zichtbaar wordt of een actie daadwerkelijk plaats heeft gevonden. Om de toevoegdatum en wijzigdatum automatisch te laten invullen heb ik twee triggers aangemaakt. Eén trigger gaat af bij het toevoegen van een record en vult daarbij de toevoegdatum met de systeem datum. De andere trigger gat af bij het wijzigen van een record en vult op dat moment de wijzigdatum met de systeem datum. Een voorbeeld van een trigger is weergegeven in figuur 33.

```
create or replace trigger update_toevoegdatum_artikel
 before insert on artikel
 for each row

declare
begin

select seq_auto_inc_artikel.nextval into :new.id from dual;

:new.toevoegdatum := sysdate;

end update_toevoegdatum_artikel;
```

Figuur 33: Trigger toevoegdatum artikel tabel

Auto increment

Met andere databases waarmee ik heb gewerkt was ik gewend om een auto increment veld te kunnen aangeven voor de primary keys. Na wat onderzoek op internet en navraag bij mijn bedrijfsmentor bleek dit niet zo te zijn. Ik heb dit opgelost door voor elke tabel, die dit nodig had, een sequence aan te maken. Een sequence genereert unieke getallen die voor primary keys gebruikt kunnen worden. Een voorbeeld hiervan is weergegeven in figuur 34. De trigger die afgaat bij het toevoegen van een record voert vervolgens een functie uit, die een nieuwe getal aan de sequence vraagt. Voor de tabel artikel ziet deze functie er als volgt uit:

select seq_auto_inc_artikel.nextval into :new.id from dual;

```
create sequence SEQ_AUTO_INC_ARTIKEL
minvalue 1
maxvalue 9999999999999999999999999999
start with 1
increment by 1
nocache;
```

Figuur 34: Sequence artikel tabel

9.2 Belangrijke onderdelen Oracle Apex

Voor het bouwen van de applicatie zijn er diverse onderdelen van Oracle Apex gebruikt. Om een duidelijk beeld te krijgen hoe de applicatie is opgebouwd zal ik hier de belangrijkste onderdelen beschrijven, dit zijn:

- Pages
- Regions
- Page process

9.2.1 Pages

Een applicatie gemaakt met Oracle Apex bestaat uit pagina's. Bij het aanmaken van een pagina kan er gekozen worden uit een aantal type pagina's zoals weergegeven in figuur 35.Binnen dit project is er het meest gebruik gemaakt van de volgende type pagina's:

Blank page

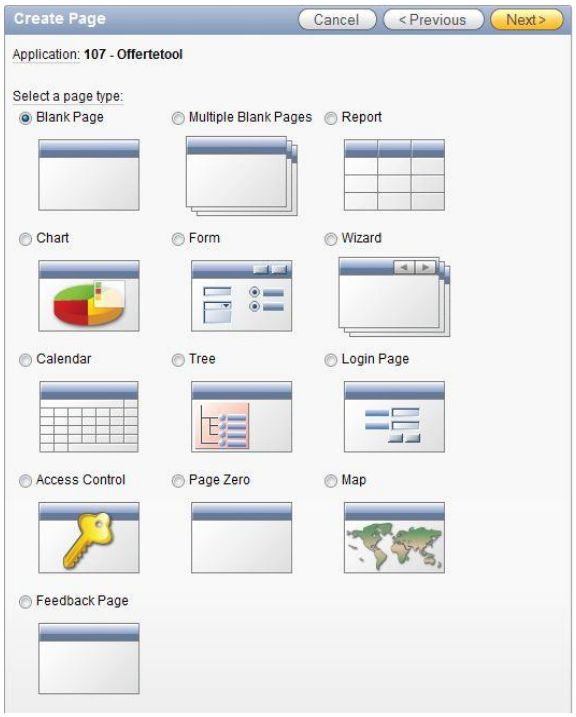
Bij de keuze van een blank page wordt de pagina alleen aangemaakt en zullen alle onderdelen op de pagina handmatig moeten worden toegevoegd.

Report page

Een report page wordt gebruikt als men een overzichtspagina wilt creëren. Binnen de applicatie voor dit project is zo'n type pagina o.a. gebruikt om een overzicht van alle artikelen te tonen. Bij het aanmaken van een report page moet een SQL query worden ingevuld voor het ophalen van de gegevens. Oracle Apex zorgt er vervolgens voor dat de gegevens in een overzicht worden getoond.

Form page

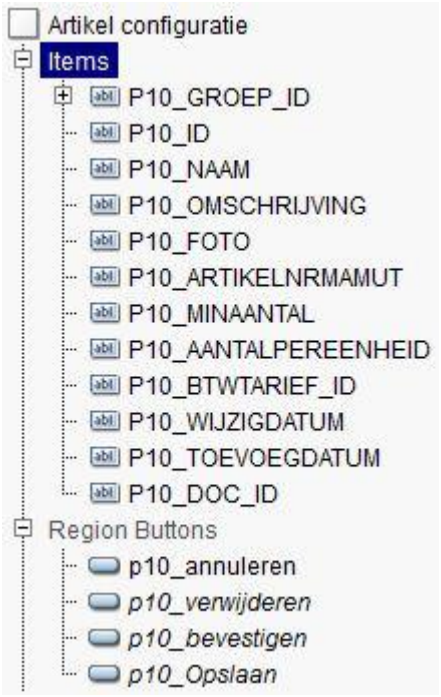
Een form page is een interessante pagina type binnen Oracle Apex. Deze pagina kan o.a. worden aangemaakt op basis van een tabel, view of een SQL query. Als er bijvoorbeeld wordt gekozen om een form page aan te maken op basis van een tabel dan moet er gekozen worden welke attributen uit een tabel in het formulier moeten worden weergegeven. Na het bevestigen van het aanmaken van de pagina creëert Oracle Apex automatisch buttons en processes om gegevens uit het formulier toe te voegen in de tabel.



Figuur 35: Pagina aanmaken

9.2.2 Regions

Een pagina binnen Oracle Apex is opgebouwd uit regions. Ook bij het aanmaken van een Region kan worden gekozen uit een aantal types zoals een Form Region, Report Region. Op deze manier kunnen bijvoorbeeld 2 report regions op 1 pagina worden aangemaakt om 2 overzichten te tonen. Een region kan bestaan uit verschillende onderdelen maar de belangrijkste om te noemen zijn Items en Buttons zoals weergegeven in figuur 36. Een page item is bijvoorbeeld een textfield, select list of password field. Een button kan worden gebruikt om een page proces te laten uitvoeren voor bijvoorbeeld het opslaan van gegevens. Een page process zal ik in de volgende paragraaf toelichten.



Figuur 3613 Region inhoud

9.2.3 Page proces

Page processes zijn er in diverse categorieën, hieronder zal ik een page process toelichten die binnen deze applicatie het vaakst gebruikt is, namelijk een PL/SQL process.

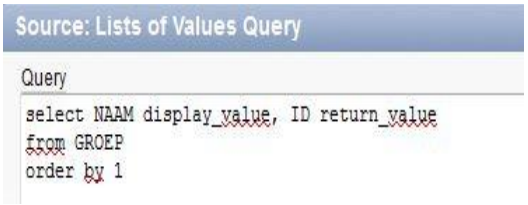
PL/SQL process

Een PL/SQL process is een process op basis van PL/SQL code voor bijvoorbeeld het ophalen of toevoegen van data. Bij het aanmaken van een page process kan men uit een aantal opties kiezen wanneer deze wordt uitgevoerd. Bijvoorbeeld bij het laden van een pagina of na het verlaten van een pagina.

9.3 Artikelen beheren

Een belangrijk onderdeel van de applicatie is het toevoegen, wijzigen en verwijderen van artikelen. Dit lijkt simpel te implementeren maar dit heeft behoorlijk veel tijd en moeite gekost. Voor de implementatie hiervan heb ik gebruik gemaakt van 2 pagina's: een report page om een overzicht van alle artikelen te tonen en een form page voor het invoeren, wijzigen en verwijderen van artikelen.

De overzichtspagina van artikelen wordt bij het creëren gebaseerd op alle attributen uit de tabel "Artikel". De tabel "Artikel" bevat enkele foreign keys naar andere tabellen bijvoorbeeld groep_id naar de tabel "Groep". Na het aanmaken van de pagina wordt in het overzicht de kolom groep_id getoond met daaronder de id van de groep, echter hier moest de naam van de groep worden getoond. Om dit te realiseren heb ik gebruik gemaakt van List of Values zoals weergegeven in figuur 37. Bij de eigenschappen van de kolom groep_id heb ik vervolgens aangegeven dat deze op basis van de List Of Value moet worden weergegeven.



Figuur 3714 List of Values Groep

Figuur 38 List of Values aangeven

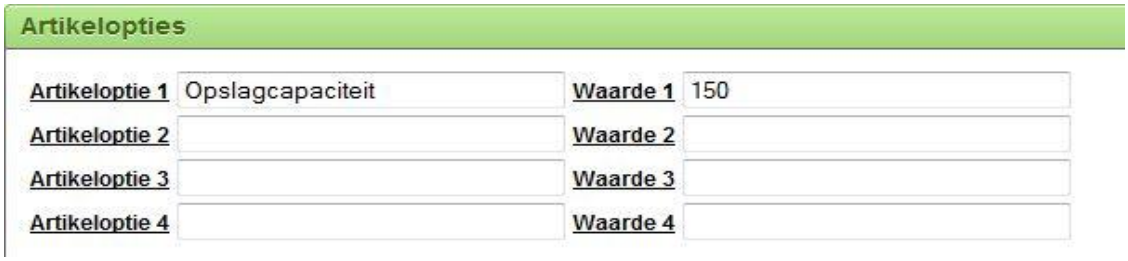
Om artikelen te kunnen toevoegen, wijzigen of verwijderen heb ik een form page aangemaakt. Deze pagina is bij het aanmaken gebaseerd op de tabel "Artikel", Apex creëert automatisch alle page items, buttons en page processes voor het toevoegen, wijzigen en verwijderen.

Op de pagina moeten ook gegevens worden ingevuld die naar andere tabellen worden weggeschreven, dit was echter lastiger om te realiseren met name het gedeelte van artikelopties en kostenopties. Om dit beter te begrijpen zal ik hier even kort het principe van artikelopties beschrijven:

- Artikelopties(bv. Resolutie, opslagcapaciteit) worden bij het toevoegen van een artikelgroep toegevoegd.
- Als men vervolgens een artikel gaat toevoegen moet geselecteerd worden in welke artikelgroep deze thuis hoort en worden de bijbehorende artikelopties getoond en hoeft alleen de waarde(bv. 150gb bij opslagcapaciteit) te worden ingevuld.

In eerste instantie wilde ik voor artikelopties een aparte region met page items aanmaken waar de waarde van artikelopties kan worden ingevuld. Echter ik liep hier tegen het probleem aan dat er een vast aantal page items moet worden aangemaakt waar gegevens kunnen worden ingevuld terwijl maar een gedeelte daarvan gebruikt wordt.

In figuur 39 bijvoorbeeld moet alleen de waarde van opslagcapaciteit worden ingevuld, artikeloptie 2,3,4 blijven dan ongebruikt. Hierdoor kan validatie van velden niet goed worden toegepast omdat niet bekend is hoeveel velden moeten worden ingevuld. Tevens levert het problemen op als er meer waardes moeten worden ingevuld dan er velden beschikbaar zijn.

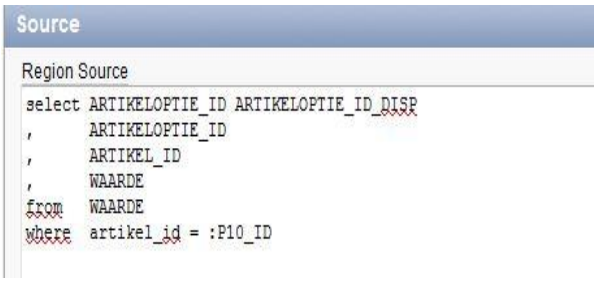


Figuur 3915: Verkeerde keuze om artikeloptie waardes in te voeren

Om hiervoor een betere oplossing voor te vinden heb ik informatie gezocht op Oracle Docs en Oracle forums, hier kwam ik het gebruik van tabular form regions tegen. Dit is een formulier gebaseerd op een SQL query en past automatisch het aantal invoer velden aan de hand van de SQL query. Dit was uitstekend geschikt om toe te passen voor het tonen van artikelopties en het invoeren van de bijbehorende waardes. Het enige obstakel hiermee is dat het “Artikel ID” bekent moet zijn om de tabel waarde te vullen echter het artikel moet nog worden ingevoerd dus dat ID is nog niet bekend, ik wilde immers met 1 button alle gegevens van de pagina toevoegen.

Mijn bedrijfsmentor gaf mij het advies om contact op te nemen met de Oracle Apex Expert die beschikbaar was om te helpen met dit soort problemen tijdens het project. In een remote sessie(hulp op afstand) heeft hij mij diverse oplossingen getoond voor het omgaan met dit soort problemen. Tevens heeft hij mij gelijk andere functionaliteiten van Oracle Apex laten zien die erg bruikbaar waren. Het belangrijkste advies was om eerst de gegevens van de tabel artikel op te slaan zodat het ID van een artikel vervolgens bekend is en deze in een “Verborgen Page Item” kan worden gebruikt. De tabular form kan vervolgens worden gemaakt op basis van de query in figuur 41.In figuur 40 is de layout van het tabular form te zien. Het proces voor toevoegen, wijzigen van de waardes wordt door Oracle Apex automatisch aangemaakt bij het aanmaken van de tabular form.

Figuur 4016 Definitieve layout tabular form artikeloptie waardes invullen



Figuur 4117 SQL Source tabular form artikeloptie waardes invullen

Het tabular form voor artikelopties werkte uitstekend en dit principe wilde ik ook toepassen voor kostentypes. Echter bij het aanmaken van een tabular form voor kostentypes gaf Oracle Apex de melding dat er maar 1 tabular form per pagina wordt toegestaan. Dit was een flinke domper en heeft me behoorlijk wat tijd gekost om hiervoor een goede oplossing te vinden.

In verschillende forums en op openbare workspaces werd verteld over het direct aanspreken van een “APEX_ITEM” package. Deze kan gebruikt worden voor het aanmaken van formulier velden die gebaseerd zijn op een SQL query. Om dit toe te kunnen passen heb ik een SQL Report region aangemaakt en in de source van de region vervolgens een SQL query geschreven die de verschillende functies van de APEX_ITEM package gebruikt. Bijvoorbeeld voor het aanmaken van invoervelden de apex_item.text() functie. In figuur 42 is de volledige source van de region te zien.

Region Source

```
SELECT apex_item.checkbox (30,
                           artikel_id,
                           'onclick="highlight_row1(this,' || ROWNUM || ')"',
                           NULL,
                           ':',
                           'f30_' || LPAD (ROWNUM, 4, '0')
                          ) delete_checkbox,
       artikel_id,  apex_item.hidden (31, artikel_id)

       || apex_item.text (32,
                          verkoopprijs,
                          80,
                          100,
                          'style="width:170px"',
                          'f32_' || LPAD (ROWNUM, 4, '0')
                         )
       ||

        apex_item.text (34,
                          inkoopprijs,
                          80,
                          100,
                          'style="width:170px"',
                          'f34_' || LPAD (ROWNUM, 4, '0')
                         )
       ||

    apex_item.display_and_save(36, kostentype_id) kostentype_id,

apex_item.text_from_lov_query (kostentype_id,'SELECT naam, id from kostentype ') Kostentype

FROM prijs  where artikel_id = :P10_ID
```

Figuur 4218: SQL Source tabular form kostentypes

Voor het toevoegen en wijzigen van kosten heb ik een PL/SQL page process aangemaakt. In deze PL/SQL functie wordt gebruik gemaakt van de apex_application.g array. Door over deze array te itereren kunnen de verschillende waardes van de invoervelden direct worden aangesproken. In figuur 43 is de PL/SQL page process weergegeven.

```
BEGIN

   FOR i IN 1 .. apex_application.g_f31.COUNT
   LOOP
      IF     apex_application.g_f33 (i) IS NOT NULL
      THEN
         UPDATE prijs
            SET verkoopprijs = apex_application.g_f32 (i), inkoopprijs = apex_application.g_f34 (i)
          WHERE artikel_id = apex_application.g_f31 (i) and kostentype_id = apex_application.g_f36(i);

      ELSIF     apex_application.g_f33 (i) IS NULL
            AND (   apex_application.g_f32 (i) IS NOT NULL
                )
      THEN
         INSERT INTO prijs
                     (test,inkoopprijs
                     )
              VALUES (apex_application.g_f32 (i), apex_application.g_f34 (i)
                     );
      END IF;
   END LOOP;

END;
```

Figuur 4319: PL/SQL page proces kosten invoeren

9.4 Offerte wizzard

De offerte wizzard voor het opstellen van offertes is opgebouwd uit een viertal pagina's:

- Startpagina
- Groepkeuze welke mee te nemen in de wizard
- Vragen & Antwoordopties pagina
- Besteloverzicht

9.4.1 Startpagina offerte wizzard

De startpagina van de offerte wizzard is opgebouwd uit 2 regions:

- Region voor prodcutindeling
- Region voor starten offerteoplossing

Region voor productindeling

Om duidelijkheid richting de klant toe te blijven behouden wordt bij de start van de offerte wizard de huidige productindeling(Productgroep -> Product) getoond zoals weergegeven in figuur 44. Hiervoor heb ik gebruik gemaakt van een Tree region. Een Tree region kan worden gemaakt op basis van een SQL query om een hiërarchie in een boomstructuur weer te geven.

Figuur 4420: Start offerte wizzard

In de SQL query, weergegeven in figuur 45, wordt bij de kolom “link” het page item P30_SELECT_NODE aangegeven. Dit is een verborgen page item waarin het ID van het geselecteerde product uit de boomstructuur wordt opgeslagen zodra deze wordt geselecteerd.

* Tree Query

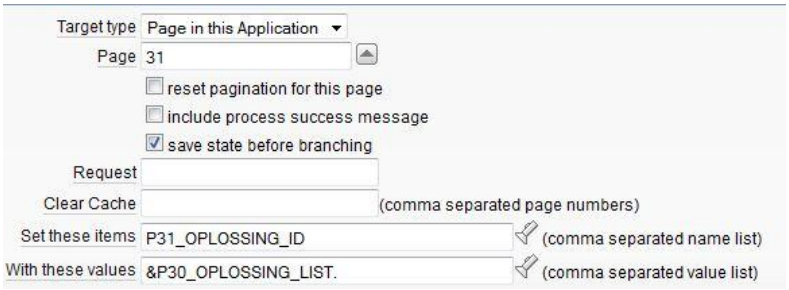
```
select case when connect_by_isleaf = 1 then 0
            when level = 1             then 1
            else                            -1
       end as status,
       level,
       "NAAM" as title,
       null as icon,
       "ID" as value,
       null as tooltip,
      'f?p=107:30:&SESSION.::::P30_SELECT_NODE:'||id as link

from categorie
start with id = 1
connect by prior "ID" = "CATEGORIE_ID"
order siblings by "NAAM"
```

Figuur 4521 Tree region source

Region voor starten offerteoplossing

De tweede region op de pagina wordt gebruikt voor het selecteren van een offerteoplossing en deze vervolgens te starten. Zodra in de bovenste region een product wordt geselecteerd toont de onderste region de bijbehorende oplossingen. In figuur 44 is bijvoorbeeld het product “IP-Camerabeveiliging” geselecteerd en in de onderste region wordt de offerteoplossing “IP-Camerabeveiliging”(toevallig zelfde naam) getoond. Als men op de start button klikt wordt de page verstuurd en een “Branch” aangesproken. De branch zorgt ervoor dat er naar een andere pagina wordt gegaan en dat een verborgen item op die pagina wordt gevuld met het ID van de geselecteerde oplossing. In figuur 46 is deze branch weergegeven.



Figuur 4622 Branch starten offerteoplossing

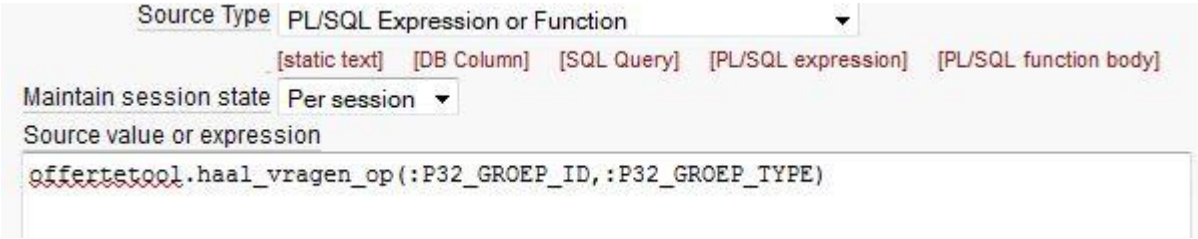
9.4.2 Vragen & Antwoordopties pagina

De vragen & antwoordopties pagina is opgebouwd uit een zestal regions. Ik hier enkele zal uitlichten, deze zijn:

- Vragen region
- Groepen antwoordopties region
- Artikel antwoordopties region

Vragen region

De vragen region bevat een “display only” page item voor het tonen van de wizard vragen. Voor het ophalen van de vragen wordt in de source van de page item een PL/SQL functie aangeroepen. Het is tevens ook mogelijk om de volledige PL/SQL functie in de source te schrijven maar qua beheer en onderhoud van de applicatie is het beter om de functie daar aan te roepen. In de functie wordt het ID en type van de gekozen groep(antwoord) meegegeven. Deze bevinden zich in twee verborgen page items en worden gevuld d.m.v. een Branch. Na een antwoord op een vraag wordt dus het ID van de gekozen groep(antwoord) in het verborgen page item “ID” gezet. Vervolgens wordt in het verborgen page item “Groep type” het type van de groep(artikelgroep of tussengroep) d.m.v. een SQL query opgehaald. In figuur 47 is de source van het page item voor het ophalen van vragen te zien waarin dus de verborgen items voor “ID” en “Groep type” worden meegegeven en figuur 48 wordt de volledige PL/SQL functie getoond.



Figuur 47 Source Page item voor tonen van vraag

```
CREATE OR REPLACE FUNCTION haal_vragen_op(groepId IN number,groepType IN NUMBER)
RETURN VARCHAR2
IS
groep_vraag varchar2(200);
BEGIN
IF groepType = 2
THEN
      SELECT vraag
      INTO groep_vraag
      FROM vraag
      WHERE groep_id = groepId;

ELSE
      SELECT vraag
      INTO groep_vraag
      FROM artikeloptie
      WHERE groep_id = groepId;

END IF;
    RETURN groep_vraag;
END;
```

Figuur 48: Functie voor vraag ophalen

Groepen antwoordopties region

Een van de voordelen van Oracle Apex is de mogelijkheid voor het instellen van condities op regions. Op deze manier kan een volledige region worden getoond of verborgen afhankelijk van bijvoorbeeld een page item. Dit heb ik toegepast op de “groepen antwoordopties region”, deze wordt namelijk getoond als het groepstype in het page item “Groep type” een tussengroep is. Op dat moment moeten namelijk de onderliggende groepen worden getoond als antwoordopties. In figuur 49 is hiervan een voorbeeld te zien. Zodra een keuze een artikelgroep is dan wordt de region verborgen en wordt de “artikel antwoordopties region” getoond.

Figuur 4923 Groepen antwoordopties

Artikel antwoordopties Region

Zoals hierboven uitgelegd wordt de “artikel antwoordopties region” getoond zodra een keuze op een vraag van het type artikelgroep is. Dit betekent namelijk dat men alle groepskeuzes heeft gehad en dat de antwoordopties geen groepen meer zijn maar artikelopties om het artikel te specificeren.

10. Evaluatie

In dit hoofdstuk evalueer ik het proces dat is gevolgd tijdens mijn afstudeerperiode en zal ik de opgeleverde producten evalueren.

10.1 Procesevaluatie

De afstudeerperiode ben ik begonnen met het onderzoeken van Oracle Apex en het bestuderen van de productcatalogus. Ik vind dat ik hier goed aan gedaan heb omdat Oracle Apex nieuw voor mij was en het van belang was om de inhoud van maar liefst 23 producten te begrijpen.

Het opstellen van de eisen en wensen vond ik in het begin best lastig. Dit kwam omdat ik niet goed wist wat er allemaal komt kijken bij het opstellen van offertes. In het eerste gesprek met de opdrachtgever kwamen er dan ook alleen standaard eisen uit zoals het beheren van artikelen en het opstellen van offertes. Door de brainstorm sessies die hierna volgden kwamen er meer en specifiekere eisen boven tafel. Ik heb deze sessies als erg prettig ervaren en waardeer ook de tijd en moeite die de opdrachtgever en bedrijfsmentor hiervoor hebben vrijgemaakt.

De planning die vooraf was opgesteld is voor driekwart precies gevolgd. Dat deze op het einde niet meer precies gevolgd kon worden kwam doordat het opstellen van het gegevensmodel langer duurde dan verwacht en door een aantal beperkingen van Oracle Apex die ik tegen kwam bij het bouwen van de applicatie.

Dat het opstellen van het gegevensmodel langer heeft geduurd dan verwacht vonden de opdrachtgever en ik niet heel erg omdat dit wel heeft geresulteerd in een model waarin rekening is gehouden met alle eisen en wensen van de opdrachtgever. Ik ben best trots op het resultaat hiervan en mijn opdrachtgever en bedrijfsmentor waren hier ook tevreden over. Van te voren was ook aangegeven dat dit het belangrijkste onderdeel van het project zou zijn.

Helaas was de opgenomen uitloopperiode niet voldoende om de genoemde vertragingen op te vangen. Hierdoor zijn niet alle functionaliteiten gerealiseerd in de applicatie en heeft er geen testfase plaats kunnen vinden waarin ik een gebruikersacceptatie test wilden uitvoeren. De opdrachtgever had hier wel begrip voor en met hem is ook de afspraak gemaakt om de laatste functionaliteiten af te maken na het inleveren van dit document.

Het gebruik van RUP is goed verlopen tijdens dit project en heeft voldoende houvast geboden om dit project uit te voeren. Door het gebruik van iteraties is de opdrachtgever nauw betrokken geweest tijdens de uitvoering van het project. Dit heeft naar mijn inziens geresulteerd dat alles wat is opgeleverd naar wens is geweest.

Terugkijkend naar de gehele afstudeerperiode ben ik best tevreden. Ik heb enorm veel geleerd over het bouwen van een database en Oracle Apex. De gesprekken met de opdrachtgever en bedrijfsmentor zijn in goede sfeer verlopen. Achteraf gezien had ik wel eerder contact moeten opnemen met de Oracle Apex expert i.p.v. alles zelf proberen uit te zoeken, hierdoor hadden problemen bij het bouwen eerder opgelost kunnen worden.

10.2 Productevaluatie

10.2.1 Plan van aanpak

Zoals in elk project is het eerste product dat gemaakt wordt het plan van aanpak, zo ook in dit project. Het plan van aanpak is een document geworden waarin alle onderwerpen zijn beschreven die te maken hebben met de uitvoering van dit project. Dit heeft niet alleen mij een goed inzicht maar ook de opdrachtgever. Het plan van aanpak heb ik dan ook een paar keer met de opdrachtgever en bedrijfsmentor besproken. De opdrachtgever was vooral geïnteresseerd in de planning, vergaderpunten en op te leveren producten en mijn bedrijfsmentor keek ook naar het gebruik van tools, methoden en technieken. Deze belichting van het document vanuit zowel opdrachtgevers oogpunt als bedrijfsmentor heb ik als erg prettig ervaren en heeft ook bijgedragen aan de kwaliteit van het document. Omdat ik al regelmatig een plan van aanpak heb opgesteld kon ik vrij snel de juiste onderwerpen opnemen. Tevens is het achteraf een goede beslissing van mij geweest om niet direct bij aanvang te starten met het maken van het plan van aanpak maar eerst kennis op te doen van o.a. Oracle Apex. Ik vind het plan van aanpak een keurig document geworden dat voldoende is geweest om het project mee te kunnen starten.

10.2.2 Vision document

Het Vision document is een document geworden waarin naar mijn inziens zeer uitgebreid de probleemstelling, belanghebbenden en een opsomming van eisen en wensen is gegeven.
Het opstellen van dit document heeft behoorlijk wat tijd en moeite gekost. Met name het benoemen van de juiste rollen van de belanghebbenden, hun behoeften en het beschrijven van de functionaliteit eisen. Dit kwam o.a. doordat het huidige administratiepakket geen verdeling kent van belanghebbenden. De opdrachtgever ging hier dus ook pas goed over nadenken toen ik hem dit vraagstuk had voorgelegd. Hij vond dit echter wel een interessant onderdeel want zo kreeg hij ook meer inzicht in zijn eigen bedrijfsproces terwijl hij normaal hier nooit zo mee bezig was.

10.2.3 Use Case Model document

Over het Use Case Model document ben ik redelijk tevreden. Ik vind dat de verschillende rollen en verantwoordelijkheden met het systeem goed zijn verwerkt in de diagrammen. De use case beschrijvingen had ik achteraf gezien specifieker kunnen uitwerken i.p.v. dat ik deze redelijk globaal had beschreven.

10.2. 4 Functioneel ontwerp

Ik vind dat in het functioneel ontwerp erg duidelijk de werking van de applicatie is weergegeven. Het ontwerpen hiervan heeft mij ook enorm geholpen om bepaalde functionaliteiten richting de opdrachtgever toe duidelijker uit te leggen. Wat achteraf beter had gekund was om het functioneel ontwerp volledig op Oracle APEX te baseren. Bij het bouwen van de applicatie is dan precies bekend hoe de functionaliteiten praktisch verwerkt kunnen worden. Hierdoor zullen de functionaliteiten ook sneller gerealiseerd kunnen worden.

10.2.5 Klassendiagram

Ik ben erg tevreden over het opgestelde klassendiagram. Bij het ontwerpen hiervan is met alle eisen van de opdrachtgever rekening gehouden. Dit heeft geresulteerd in een enorm uitgebreid klassendiagram, dat ik van tevoren zeker niet had verwacht. Een groot verschil met de “school cases” is dat er door de bedrijfsmentor en opdrachtgever veel gehamerd wordt op onderhoudbaarheid en uitbreidbaarheid, dit is iets dat ik niet zo gewend was. Dit zijn tijdens het project voor mij leerpunten geweest om rekening mee te houden.

10.2.6 Relationeel Representatiemodel

Ik vind dat het relationeel representatiemodel een prima model geworden, waarbij ik goed heb nagedacht over het vertalen van het klassendiagram naar de toekomstige tabellen. Omdat het klassendiagram al redelijk specifiek was uitgewerkt kon ik dit model ook gemakkelijker opstellen.

10.2.7 Relationeel Implementatiemodel

Ik ben erg blij dat ik voor het opstellen van het implementatiemodel Jdeveloper heb gebruikt, dit heeft mij enorm veel tijd bespaart. De implementatie van dit model verliep hierdoor ook erg soepel.

10.2.8 Offertetool

Ik vind het erg jammer dat het eindresultaat niet volledig gerealiseerd kon worden in de beschikbare tijd. Achteraf gezien had er meer tijd moeten worden gereserveerd voor het bouwen van de applicatie. Het bouwen van de applicatie ging met Oracle Apex niet zo snel als ik had verwacht. Desondanks is het beheren van artikelen, artikelgroepen, tussengroepen, producten en productgroepen gerealiseerd en ook getest tijdens het bouwen. Ook kan er een offerteoplossing worden gestart en werkt het tonen van vragen en antwoordopties. Alleen de laatste stap van het kiezen van een artikel en daarvan een offerte te genereren ontbreekt nog.

11. Afwijkingen ten opzichte van afstudeerplan

In het afstudeerplan, zie de bijlage hiervoor, is een opsomming van op te leveren producten opgesteld. Tijdens mijn oriëntatiefase ben ik erachter gekomen dat hier onjuiste bewoordingen zijn gebruikt. Tevens heb sommige producten anders verwerkt. Ik zal hier een opsomming geven van producten die zijn afgeweken ten opzichte van het afstudeerplan:

- Pakket van eisen en wensen
  Bij het bestuderen van informatie over RUP vond ik het een beter idee om de eisen en wensen te verwerken in het vision document. Hierdoor is er 1 document waarin de gehele visie tussen opdrachtgever en opdrachtnemer is verwerkt.
- Technisch ontwerp applicatie
  Voor Oracle Apex bleek geen technisch ontwerp nodig zijn zoals bijvoorbeeld een klassendiagram. Om mezelf wel te ondersteunen bij de technische implementatie van de applicatie is een activitydiagram opgesteld
- Functioneel en technisch ontwerp database
  Dit zijn onjuiste bewoordingen geweest in het afstudeerplan want deze termen komen bij het ontwerpen van databases niet voor. Ik heb de ontwerpen van de database verwerkt in het gegevensmodel, relationeel representatiemodel en relationeel implementatiemodel

12. Aantonen beroepstaken

12.1 Uitvoeren analyse door definitie van requirements

Ik vind dat ik voldaan heb aan deze beroepstaak. In de inception fase heb ik meerdere gesprekken met de opdrachtgever en bedrijfsmentor gevoerd om de eisen en wensen op te stellen. Hiervoor heb ik gebruik gemaakt van het ISO 9126 model zodat met alle kwaliteitseisen van software rekening is gehouden. Alle opgestelde eisen en wensen zijn verwerkt in het Vision document waarna deze geprioriteerd zijn aan de hand van de MoSCoW-analyse.

12.2 Opstellen gegevensmodel voor een database

Het opstellen van een gegevensmodel voor een database is een enorme uitdaging geweest. Hiervoor is er gebruik gemaakt van het klassendiagram waarin rekening is gehouden met alle eisen van de opdrachtgever. De groepenstructuur, zoals uitgelegd in hoofdstuk 7, heeft het mogelijk gemaakt om via wizard gestuurde vragen een offerte op te stellen. Maar dit betekende ook dat er met nog meer zaken rekening moest worden gehouden bij het opstellen van het gegevensmodel. Bijvoorbeeld documenten hebben niet alleen nog maar een relatie met een artikel maar ook met een groep. Artikelopties hebben niet alleen een relatie meer met een artikel maar ook met een artikelgroep, zodat het toevoegen van nieuwe artikelen gemakkelijk blijft. Tevens is er goed nagedacht over uitbreidbaarheid door een aparte tabel voor kostentypes op te nemen. Alle bovengenoemde en alle overige eisen van de opdrachtgever zijn verwerkt in 1 compleet gegevensmodel. Ik vind dat ik hiermee heb aangetoond te voldaan aan deze beroepstaak.

12.3 Ontwerpen, bouwen en bevragen van een database

Het vertalen van het gegevensmodel naar een relationeel implementatiemodel heeft een flinke bijdrage geleverd aan het behalen van deze beroepstaak. Hierbij heb ik o.a. rekening gehouden met de keuzes van datatypes die Oracle kent. Ook heb ik gedacht aan de performance door de verschillende groepen om te zetten naar 1 tabel zodat er minder joins nodig zijn. Tevens heb ik meerdere triggers geschreven om aanpassingen op de database te kunnen traceren en heb ik PL/SQL functies geschreven voor het bevragen van de database en hergebruik te waarborgen.

12.4 Bouwen applicatie

Ik vindt dat ik voldaan heb aan deze beroepstaak omdat ik de gehele applicatie met Oracle Apex heb gebouwd. Dit betekende dat ik alle functionaliteiten zelf moest realiseren en hierbij heb ik de mogelijkheden van Oracle Apex volledig benut. Ook heb ik zelf keuzes moeten maken in o.a. pagina opbouw, het gebruik van page processes en in het oplossen van beperkingen die Oracle Apex met zich mee brengt.

13. Literatuurlijst

1. Remi-Armand Collaris & Eef dekker, Rup Op maat(2006), ISBN 90 12 11413 6

2. Jos Warmer & Anneke Kleppe(2006), Praktisch UML(2006), ISBN 978-90-430-1265-2

3. http://docs.oracle.com, Oracle

4. http://www.rupopmaat.nl, Rup Op Maat