Gestructureerde documentatie van functioneel ontwerp

Een modelgebaseerde aanpak van functioneel ontwerpdocumentatie

Masterscriptie

Dennis Janse
Afstudeernummer: 596

19 november 2008

Onder begeleiding van:
Patrick van Bommel (Radboud Universiteit Nijmegen)
Geert Vissers (Radboud Universiteit Nijmegen)
Jacob Kleerekoper (Quinity BV)
Jeroen Snijders (Quinity BV)
Robert Guitink (Quinity BV)

Samenvatting

Een onderdeel in het ontwikkelen van een informatiesysteem is het functioneel ontwerp. Dit is een belangrijk communicatiemiddel tussen klant, ontwerpers en ontwikkelaars. Er zijn vele manieren om functioneel ontwerp te documenteren. In deze scriptie is beschreven hoe dit bij Quinity gebeurt. Gebaseerd op praktijkonderzoek worden een aantal eisen geformuleerd waaraan functioneel ontwerpdocumentatie dient te voldoen. Die eisen zijn: een goede onderhoudbaarheid, traceerbaarheid van requirements, opname van annotaties en intelligente zoekmogelijkheden binnen functioneel ontwerpdocumentatie. In drie hedendaagse systeemontwikkelingsmethoden komen deze eisen om het functioneel ontwerp documenteren ook voor. Op basis van dit praktijk- en literatuuronderzoek stel ik een documentatiewijze voor waarmee de eisen aan functioneel ontwerpdocumentatie kunnen worden vervuld. De documentatiewijze beschrijft een functioneel ontwerpmodel op basis waarvan functionele ontwerpen kunnen worden opgeslagen. Het model kan worden gebruikt voor de implementatie of keuze van tools voor functioneel ontwerpdocumentatie. Hiervan zijn een aantal gebruiksmogelijkheden beschreven.

Dankwoord

In dit dankwoord wil ik graag de mensen bedanken die op welke wijze dan ook betrokken zijn geweest bij mijn onderzoek. Allereerst dank ik Quinity voor de mogelijkheid om mijn afstudeeronderzoek binnen het bedrijf uit te voeren. In het bijzonder dank ik Jacob Kleerekoper voor de dagelijkse begeleiding, de opbouwende kritiek en de aanmoediging. Daarnaast bedank ik ook Jeroen Snijders en Robert Guitink voor de input tijdens de tweewekelijkse overlegbijeenkomsten. Ook dank ik de (klant)medewerkers die input hebben geleverd via interviews, gesprekken of op andere wijzen.

Verder dank ik de begeleiding vanuit de Radboud Universiteit Nijmegen. In het bijzonder dank ik Patrick van Bommel voor de goede vragen en opbouwende kritiek tijdens de maandelijkse voortgangsbijeenkomsten. Ook Geert Vissers wil ik hartelijk danken voor het gedegen commentaar, in het bijzonder op het praktijkonderzoek.

Als laatste wil ik mijn vrouw Christine bedanken voor haar voortdurende steun, betrokkenheid en geduld, vooral wanneer het onderzoek even tegenzat.

Inhoudsopgave

1. Inleiding

Quinity maakt software voor de markt van financiële dienstverlening. Het bedrijf levert daarvoor de Quinity Insurance Solution (QIS), een verzameling standaardcomponenten voor verzekeringsadministraties. Voordat de software geleverd kan worden, is een uitgebreid communicatieproces nodig om af te stemmen wat de klant van de software verwacht. In het algemeen is het erg lastig om te documenteren wat de software moet doen en hoe die software moet werken. Die documentatie is niet alleen van belang voor de communicatie tussen klant en ontwikkelaar, maar ook voor communicatie tussen de medewerkers van de ontwikkelorganisatie zelf. In de softwareindustrie falen nog steeds veel softwareprojecten als gevolg van gebrekkige communicatie. Een project wordt echter niet altijd volledig afgeblazen. Het komt vaker voor dat softwareprojecten over het budget heengaan of te laat worden opgeleverd. Niet alles is te wijten aan gebrekkige communicatie, maar dit is wel een belangrijke oorzaak daarin. Al jaren is er onderzoek naar verbetering van de communicatie in softwareprojecten. Naast verbeteringen in methoden zijn er ook verbeteringen in ondersteunende hulpmiddelen, modelleringstechnieken en best practices. Dit onderzoek richt zich op de documentatie die in de functioneel ontwerpfase[1] wordt vastgelegd. In dit hoofdstuk introduceer ik het onderwerp van mijn onderzoek. Eerst zal ik een beeld schetsen van de branche waarin Quinity opereert en hoe het bedrijf daarin werkt. Dan zal ik de belangrijkste begrippen binnen dit onderzoek toelichten. Vervolgens zal ik een overzicht geven van eerder onderzoek binnen Quinity. Als laatste laat ik zien wat ik in deze scriptie ga behandelen.

1.1. Context

De applicaties die Quinity maakt zijn voornamelijk voor de financiële sector. Veel van die applicaties worden op maat gemaakt op basis van standaardcomponenten. Hiervoor levert Quinity de Quinity Insurance Solution (QIS). Dit is een "*volledig geïntegreerde webbased polis- en schadeadministratie voor verzekeringsmaatschappijen, volmachten en tussenpersonen ... om de front-, mid- en backofficeprocessen van een polis en een schadedossier te ondersteunen*"[2]. QIS bestaat uit een aantal componenten die een klant los kan afnemen. Het gebruik van deze componenten zorgt voor hogere kwaliteit en meer functionaliteit tegen lagere kosten en een snellere doorlooptijd voor nieuwe projecten. Door het hergebruik van de componenten door meerdere klanten kunnen de componenten telkens worden verbeterd en uitgebreid met nieuwe functionaliteit. Nieuwe projecten kunnen dan gebruik maken van de reeds bestaande functionaliteit die de standaardcomponenten al bieden.

[1] In bijlage A is een woordenlijst opgenomen waarin veelgebruikte woorden in dit onderzoek worden uitgelegd (zoals 'functioneel ontwerp'). Paragraaf 1.2 beschrijft overigens ook al een aantal begrippen.

[2] Zie website Quinity: `http://www.quinity.com/`

In het ontwikkelproces van een informatiesysteem is het ontwerp een belangrijk onderdeel. Eerst stelt een ontwerper een functioneel ontwerp op dat specificeert hoe het systeem de requirements uitwerkt en wat de gebruiker er mee kan doen. Het functioneel ontwerp bestaat uit een functionele beschrijving van het te bouwen systeem, een datamodel dat de gegevens van de applicatie en de samenhang tussen die gegevens weergeeft en een prototype dat de interactie van gebruikers beschrijft. Na het functioneel ontwerp komt het technisch ontwerp dat beschrijft hoe de functionaliteit gerealiseerd moet worden. Hierna kan de applicatie geïmplementeerd worden in programmacode. De realisatie van een applicatie wordt bereikt door een gestructureerde werkwijze, waarbij er veel aandacht is voor het functioneel ontwerp van de applicatie. Dit ontwerp is ook een belangrijk communicatiemiddel in het contact met de klant. Voor de verschillende fasen in het ontwikkelproces zijn er standaarden en richtlijnen om structuur te geven aan de activiteiten die worden uitgevoerd en de producten die worden opgeleverd. Dit onderzoek richt zich op de producten die tijdens de functioneel ontwerpfase worden opgeleverd: de functioneel ontwerpdocumentatie.

1.2. Conceptueel model

In deze scriptie speelt een aantal begrippen een belangrijke rol. Hieronder een lijstje van de definities van de belangrijkste begrippen.

Informatiesysteem Een stelsel van gegevens, informatiesysteemfuncties en eventueel systeemgebonden technische structuur. Het doel ervan is: het verzamelen, vastleggen, verwerken, bewaren, transporteren en verstrekken van gegevens ten behoeve van het nemen van beslissingen (door mensen, programma's of apparatuur) op een afgebakend toepassingsgebied of ten behoeve van een bepaald bedrijfsdoel. (Fokkinga et al., 2002)

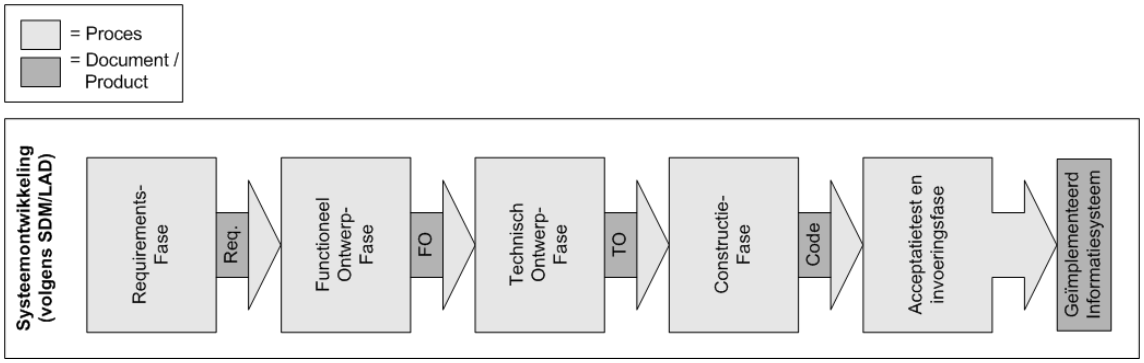
Systeemontwikkeling Het proces waardoor een Informatiesysteem tot stand komt. Vaak is dit proces opgedeeld in een aantal fasen.

Klant Een persoon of groep personen die de eisen en wensen over het informatiesysteem specificeert.

Requirement Een eis of wens van de klant over het te bouwen systeem. Er is onderscheid tussen *functionele* en *non-functionele* requirements. Een functionele requirement is een eis of wens over de functionaliteit van het systeem. Een non-functionele requirement is een eis of wens over de randvoorwaarden waarmee het systeem om moet gaan.

Functioneel ontwerp Een beschrijving van gegevens in een informatiesysteem, gedrag van een informatiesysteem en de manier waarop gebruikers kunnen interacteren met een informatiesysteem. Deze beschrijving is zodanig dat klantmedewerkers kunnen controleren wat de klant krijgt en dat ontwikkelaars het informatiesysteem kunnen implementeren.

Technisch ontwerp Een beschrijving van de technische ontwerpkeuzes en de wijze waarop de functionaliteit uit het functioneel ontwerp is geïmplementeerd in de code. Een functioneel ontwerp kan technisch op meerdere manieren worden geïmplementeerd. Een technisch ontwerp is een verantwoording en een beschrijving van die keuzes.



Figuur 1.1.: Conceptueel Model

In figuur 1.1 zijn de begrippen en de onderlinge relaties weergegeven. Systeemontwikkeling begint met de requirementsfase waarin de requirements worden opgesteld. De laatste fase is de acceptatietest en invoeringsfase. Deze leidt tot een in de organisatie geïmplementeerd informatiesysteem. De klant kan betrokken zijn bij alle fasen, zij het dat die betrokkenheid in de technisch ontwerp en constructiefase vaak minder en anders zal zijn dan in de andere fasen. In de eerste twee fasen zal de klant vooral zijn eisen en wensen kenbaar maken, in de twee fasen die daarop volgen zal de klant vooral betrokken zijn als vraagbaak voor dingen die nog niet duidelijk waren en in de laatste fase zal de klant vooral een controlerende rol spelen waarin hij[3] kenbaar maakt of het informatiesysteem aan zijn eerder geformuleerde eisen en wensen voldoet.

1.3. Eerder onderzoek

Om nieuwe applicaties sneller en goedkoper te ontwikkelen, is het handig om eerdere vindingen opnieuw te gebruiken. De kwaliteit van die applicaties kan ook worden verhoogd door gebruik te maken van best practices. Dit wordt voor programma's, code en technische ontwerpen al geruime tijd gedaan. Quinity doet sinds 2002 onderzoek naar hergebruik van functionele ontwerpen. Sinds 2004 zijn er ook afstudeerders bezig met onderzoek naar hergebruik van functionele ontwerpen en daarmee samenhangende onderwerpen. Een overzicht van de resultaten uit dit onderzoek is hieronder weergegeven.

- 2004, augustus: Jeroen Snijders, definitie van het begrip 'functioneel ontwerppatroon'. Daarnaast gaat hij in op de wijze van beschrijven en gebruiken van dit type patroon. Ook de verhouding van functionele ontwerppatronen tot andere technieken voor hergebruik komt aan bod (Snijders, 2004).
- 2005, december: Jeffrey van Helden en Niels Reyngoud, verdieping en uitbreiding van het begrip 'functioneel ontwerppatroon'. Daarnaast ook over het concreet maken van een ontwerppatroon en hoe ontwerppatronen toegepast kunnen worden in softwareontwikkeling (met name in het functioneel ontwerp) (van Helden en Reyngoud, 2005).

[3]Hier had ook 'zij' of 'hij of zij' gebruikt kunnen worden. Overal in de scriptie waar een soortgelijke situatie voorkomt, kan de lezer ook één van deze beide opties lezen in plaats van 'hij'.

- 2006, juli: Jeroen van Montfort, gaat in op de implementatie van functionele ontwerppatronen in technische ontwerppatronen (van Montfort, 2006).
- 2006, augustus: Peter Nagel, gaat in op het gebruik van meta-patronen. Deze meta-patronen beschrijven hoe je categorieën van functionele ontwerppatronen kunt beschrijven (Nagel, 2006).
- 2007, augustus: Jeroen van Steenbergen, over Rich Internet Applications en User Interface Patronen (van Steenbergen, 2007).
- 2007, augustus: Wouter van de Molengraft, over het gebruik van performance patronen bij relationele databases (van de Molengraft, 2007).
- 2007, augustus: Yoran Bosman, over hoe functionele ontwerppatronen gebruikt kunnen worden in het softwareontwikkelingsproces (Bosman, 2007).
- 2007, november: Jacob Kleerekoper, definieert een taal die functionele ontwerppatronen kan beschrijven (Kleerekoper, 2007).
- 2008, juli: Maarten Mulders, beschrijft een performancepatroon en beschrijft hoe performancepatronen in het algemeen gebruikt kunnen worden in het softwareontwikkelingsproces (Mulders, 2008).

Tot nu toe is er dus vooral onderzoek gedaan naar hergebruik van functionele ontwerpen. Dat hergebruik gebeurt binnen Quinity met functionele ontwerppatronen. Deze patronen beschrijven herbruikbare functionele onderdelen van toepassingen, zoals bijvoorbeeld beschrijving van loginfunctionaliteit. Documentatie van functionele ontwerppatronen is de manier om kennis vast te leggen over functioneel ontwerp. Bij het hergebruik draait het om de vraag wat voor informatie je documenteert over een oplossing voor een functioneel probleem.

1.4. Overzicht van dit onderzoek

Dit onderzoek gaat in op functionele ontwerpen zélf. Het doel is om te komen tot een verbeterplan voor de functionele ontwerpdocumentatie van Quinity. Bij de communicatie over een te bouwen applicatie is het van belang om te weten wát je wilt overbrengen en naar welke doelgroepen. Vervolgens kun je ingaan op de vraag hóé je dat gaat overbrengen naar de verschillende doelgroepen. Deze scriptie richt zich met name op de vraag wát je wilt overbrengen (en dus documenteren). Hierbij is het wel belangrijk dat je de verschillende doelgroepen van die communicatie voor ogen houdt, omdat je in hun (informatie)behoeften wilt voorzien.

De probleemstelling en de vraagstelling zijn beschreven in hoofdstuk 2, 'Probleemstelling en vraagstelling'. In dat hoofdstuk staat ook de aanpak van het onderzoek beschreven. Daarna komt hoofdstuk 3 over 'Functioneel ontwerp bij Quinity'. Hierin is de wijze beschreven waarop Quinity haar functioneel ontwerp documenteert, door het te modelleren in een UML-model. Dan volgt een hoofdstuk over het praktijkonderzoek dat ik heb uitgevoerd. Op basis van kwalitatief onderzoek binnen het bedrijf heb ik in kaart gebracht wat er aan de functioneel ontwerpdocumentatie verbeterd kan worden. In het hoofdstuk beschrijf ik de aanpak van het praktijkonderzoek, en, naar aanleiding van de resultaten en analyses van het praktijkonderzoek, de eisen waaraan de functioneel ontwerpdocumentatie dient te voldoen. Dan komt in hoofdstuk

5 functioneel ontwerp in drie hedendaagse methoden aan bod. Daarvoor komen een drietal veelgebruikte systeemontwikkelingsmethoden aan de orde. De scriptie sluit af in hoofdstuk 6 met een voorstel voor vernieuwde functioneel ontwerpdocumentatie en aanbevelingen voor verder onderzoek.

In het document 'Vertrouwelijke bijlagen Behorende bij de masterscriptie "Gestructureerde documentatie van functioneel ontwerp"' zijn de uitwerkingen en analyses van interviews met functioneel en technisch ontwerpers te vinden. Dit is een vertrouwelijk document, maar is op aanvraag in te zien door medewerkers van Quinity of externen die een vertrouwelijkheidsverklaring hebben afgelegd.

2. Probleemstelling en vraagstelling

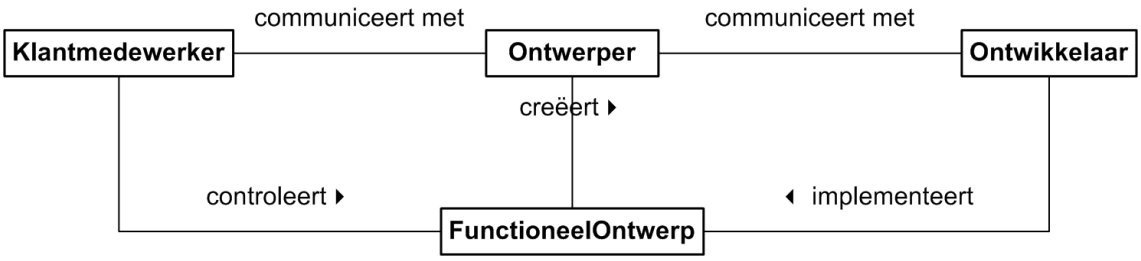
Voordat de probleemstelling geformuleerd wordt, is het belangrijk om eerst te verhelderen wat het begrip 'functioneel ontwerp' omvat, aangezien dit een kernbegrip is in dit onderzoek.

Functioneel ontwerp is aan de ene kant op te vatten als een proces en aan de andere kant als een product van dat proces. Functioneel ontwerp als product noem ik ook wel de 'functioneel ontwerpdocumentatie'. Bij het functioneel ontwerpproces is een aantal groepen betrokken. Het product 'functioneel ontwerp' komt in dat proces tot stand en is bedoeld voor verschillende doelen en groepen. Om functioneel ontwerp (als product en als proces) beter te begrijpen moet dus in kaart gebracht worden wie er betrokken is bij het opstellen van het functioneel ontwerp, voor welke groepen het functioneel ontwerp bestemd is en wat de doelen van die groepen zijn: waarvoor hebben zij het product 'functioneel ontwerp' nodig?

In het algemeen zijn er drie groepen te onderscheiden bij het ontwikkelen van een informatiesysteem. In ieder geval is er een groep die eisen en wensen formuleert (de klantmedewerkers) en een groep die deze eisen en wensen implementeert (de ontwikkelaars). Vaak zit er nog een derde derde groep tussen, die de eisen en wensen van de klantmederwerwers documenteert (de ontwerpers). Deze groepen, hun onderlinge interactie en de interactie met het functioneel ontwerp zijn weergegeven in figuur 2.1. De groepen zijn:

- **Klantmedewerker**s: deze formuleren de (informatie)behoefte over het informatiesysteem namens de klant. De klantmedewerkers communiceren deze behoeften naar de ontwerpers. Daarnaast controleren de klantmedewerkers of de ontwerpers in het functioneel ontwerp hebben beschreven wat de klant daadwerkelijk wenst.
- **Ontwerper**s: deze beschrijven de behoeften van de klant zo goed mogelijk in een functioneel ontwerp. Dit doen zij zodanig dat de klantmedewerkers het functioneel ontwerp kunnen controleren en dat ontwikkelaars weten wat ze moeten implementeren.
- **Ontwikkelaar**s: deze bouwen het daadwerkelijke informatiesysteem op basis van het functioneel ontwerp dat de ontwerpers hebben opgesteld. Als er communicatie nodig is over het informatiesysteem, overleggen de ontwikkelaars met de ontwerpers en niet direct met de klantmedewerkers.

Binnen deze groepen is nog wel een gedetailleerdere indeling mogelijk, maar voor dit globale overzicht zijn deze drie groepen voldoende. Binnen klantmedewerkers zijn bijvoorbeeld gebruikers en organisatieontwerpers te onderscheiden. Beide groepen zijn weer in andere delen van het functioneel ontwerp geïnteresseerd. In dit onderzoek gaat het over interactie tussen ontwerpers en ontwikkelaars. Daarin zijn wel meer categorieën te onderscheiden. Een ontwikkelaar kan bijvoorbeeld uitsluitend met technisch ontwerp bezig zijn, of juist uitsluitend met programmeren van code.



Figuur 2.1.: Betrokkenen bij het functioneel ontwerp

Een informatiesysteem wordt altijd ontwikkeld met het doel in een bepaalde (informatie)behoefte te voorzien. Een klant is meestal niet in staat om zelf een systeem te ontwikkelen. En zelfs al zou een klant daartoe wél in staat zijn, dan is het meestal nog niet voldoende duidelijk wat de behoeften zijn of de behoeften zijn te complex om in één keer te vertalen naar een geschikt informatiesysteem. Als er bovendien meerdere klantmedewerkers namens de klant de behoeften kenbaar maken, kunnen die behoeften elkaar ook tegenspreken. De een wil dan een functie in het systeem die een ander juist niet wil. Het is dus van belang om in kaart te brengen wat de behoeften voor het gewenste informatiesysteem zijn, zodat men overeenstemming en inzicht bereikt over de behoeften. De klantmedewerkers moeten deze behoeften op een effectieve manier kunnen communiceren naar de ontwikkelaars van het informatiesysteem. Bij deze communicatie is meestal een derde groep betrokken, dit zijn de ontwerpers[1]. Zij vertalen de klantwensen naar een beschrijving die voor klantmedewerkers, ontwikkelaars en ontwerpers duidelijk maakt welke functies er in het systeem moeten komen en hoe die functies werken. Het is een beschrijving van gegevens in, gedrag van en interactie met het informatiesysteem. De beschrijving wordt in verschillende systeemontwikkelingsmethoden anders genoemd, zoals 'functionele specificatie', 'functioneel model','software requirements specification' of 'functioneel ontwerp'. Ik zal 'functioneel ontwerp' gebruiken om dit soort ontwerp aan te duiden.

De ontwerpers proberen door het maken van het functioneel ontwerp te laten zien hoe zij denken dat het gewenste informatiesysteem in de behoeften van de klant kan voorzien. Deze terugkoppeling van ontwerpers naar de klant is nodig om onduidelijkheden en onvolledigheden in de eerdere communicatie te ontdekken en het functioneel ontwerp daar zo nodig op aan te passen. Naast deze 'foutherstelling' van de communicatie zal de confrontatie van de klant met het functioneel ontwerp van de ontwerpers waarschijnlijk óók voor de klant meer inzicht geven in wat zijn behoeften werkelijk zijn. Overigens wordt er soms ook gebruik gemaakt van een prototype dat de functionaliteit van het te bouwen inforamtiesysteem demonstreert.

Het functioneel ontwerp heeft voor de klant dus als doel dat hij weet welke functies het systeem zal krijgen en hoe die functies werken. Op die manier kan het ook dienen als contract tussen klant en opdrachtnemer. Wanneer er overeenstemming is over de werking van het systeem vanuit het gezichtspunt van de klant, kunnen de ontwikkelaars verder met de constructie van het informatiesysteem. Een ontwerper dient het systeem dus zó te beschrijven dat het ook te bouwen is. Soms is het nuttig dat een ontwerper technische opmerkingen maakt over hoe iets mogelijkerwijs geïmplementeerd kan worden. Daarnaast kan het nuttig zijn als de

[1]Bij Quinity zijn dit de functioneel ontwerpers.

ontwerper voor kan documenteren waarom het systeem moet werken zoals het beschreven is in het ontwerp. Voor de ontwerpers zelf kan dat een geheugensteun zijn en andere ontwerpers zullen beter begrijpen waarom het systeem is ontworpen zoals het is ontworpen. Het functioneel ontwerp is dus niet alleen bedoeld voor de klant, maar ook voor ontwikkelaars en ontwerpers zelf. Het kan dus zijn dat er in het functioneel ontwerp informatie wordt opgenomen die niet voor alle doelgroepen van het functioneel ontwerp nodig is.

Een functioneel ontwerp is dus een communicatiemiddel tussen klantmedewerkers, ontwerpers en ontwikkelaars. Het is belangrijk om de juiste modelleringstechniek te gebruiken voor het doel van dit communicatiemiddel, zodat de verschillende betrokkenen de kennis krijgen die ze nodig hebben (Hoppenbrouwers et al., 2005). Die drie groepen hebben allemaal verschillende doelen. Een ontwerper creëert het functioneel ontwerp zodanig dat het voor alle doelgroepen voldoet. De klantmedewerker controleert het functioneel ontwerp: "Is dit inderdaad wat ik wil". En de ontwikkelaar creëert het informatiesysteem door het functioneel ontwerp te implementeren: "Wat moet ik maken?". Ook is er in een functioneel ontwerp een duidelijk onderscheid in beschrijvingen van gegevens in, gedrag van en interactie met het informatiesysteem.

2.1. Probleemstelling

De functioneel ontwerpdocumentatie van veel systemen die Quinity maakt, neemt steeds meer toe in omvang. Hierdoor raakt het steeds lastiger het overzicht van die documentatie te bewaren. Deze situatie speelt met name bij de documentatie van de standaardcomponenten. Hoewel er standaarden en richtlijnen zijn voor het opstellen van de functioneel ontwerpdocumentatie, zorgen die er niet voor dat de documentatie overzichtelijk genoeg blijft. Graag zou Quinity een methode hebben waarmee de functioneel ontwerpdocumentatie over de systemen eenvoudig en helder kan worden vastgelegd. Het gaat er daarbij om om zo min mogelijk dubbel vast te leggen en zoveel mogelijk documentatie te hergebruiken. Daarnaast moet de documentatie begrijpelijk blijven voor de betrokkenen van functioneel ontwerp: klantmedewerkers, functioneel ontwerpers en technisch ontwerpers.

2.2. Onderzoeksvragen

1. Welke methode gebruikt Quinity nu voor functioneel ontwerpdocumentatie, en hoe wordt die methode door de betrokkenen gewaardeerd?
    a) Wat voor onderwerpen worden er in de functioneel ontwerpdocumentatie van Quinity beschreven en hoe hangen deze onderwerpen met elkaar samen?
    b) In welke opzichten voldoet de functioneel ontwerpdocumentatie aan de wensen en behoeften van elk van de betrokkenen van functioneel ontwerp en in welke opzichten niet?
        i. Wat zijn de wensen en behoeften van technisch ontwerpers met betrekking tot de functioneel ontwerpdocumentatie, en in welke opzichten voldoet Quinity's functioneel ontwerpdocumentatie aan die wensen en behoeften en in welke opzichten niet?

ii. Wat zijn de wensen en behoeften van functioneel ontwerpers met betrekking tot de functioneel ontwerpdocumentatie, en in welke opzichten voldoet Quinity's functioneel ontwerpdocumentatie aan die wensen en behoeften en in welke opzichten niet?

iii. Wat zijn de wensen en behoeften van klanten met betrekking tot de functioneel ontwerpdocumentatie, en in welke opzichten voldoet Quinity's functioneel ontwerpdocumentatie aan die wensen en behoeften en in welke opzichten niet?

2. Hoe verhoudt zich de methode van Quinity voor functioneel ontwerpdocumentatie tot hedendaagse inzichten met betrekking tot functioneel ontwerpdocumentatie?

   a) Wat zijn de eisen waaraan Quinity's functioneel ontwerpdocumentatie dient te voldoen?

   b) Welke ideeën uit andere systeemontwikkelingsmethoden kunnen worden gebruikt in Quinity's functioneel ontwerpdocumentatie?

3. Welke hedendaagse inzichten zijn van belang om Quinity's methode voor functioneel ontwerpdocumentatie zo aan te passen dat aan de behoeften en wensen van de diverse doelgroepen tegemoet wordt gekomen?

   a) Welke aspecten van functioneel ontwerpdocumentatie komen met name in aanmerking voor aanpassing?

   b) Welke concrete aanpassingen kunnen worden gedaan voor elk van die aspecten, gelet op hedendaagse inzichten met betrekking tot functioneel ontwerpdocumentatie?

2.3. Aanpak van het onderzoek

De gegevens die ik voor de beantwoording van de onderzoeksvragen nodig heb haal ik uit een viertal bronnen. Dit zijn:

1. Literatuuronderzoek: hoofdstuk 5 beschrijft verschillende systeemontwikkelingsmethoden. In ieder van die methoden komt een fase voor waarin een functioneel ontwerp (of daarmee vergelijkbare documentatie) wordt opgesteld. Het hoofdstuk geeft een overzicht en een vergelijking van de onderwerpen die de methoden beschrijven in het functioneel ontwerp. Daarnaast geeft het hoofdstuk een overzicht van de wijze waarop de onderwerpen in functioneel ontwerp worden gedocumenteerd. Deze ideeën vormen (mede) de theoretische basis van het voorstel voor functioneel ontwerpdocumentatie in hoofdstuk 6.
2. Praktijkonderzoek: door te spreken met verschillende werknemers vorm ik een beeld van de wijze waarop functioneel ontwerpdocumentatie wordt opgesteld en wat de wensen en behoeften met betrekking tot de functioneel ontwerpdocumentatie zijn. Dit zal gaan op informele wijze en in tweewekelijkse overlegbijeenkomsten met mijn begeleiders.
3. Praktijkonderzoek: topic-interviews brengen in kaart wat de taken van functioneel en technisch ontwerpers rond het functioneel ontwerp zijn. Daarnaast gaan de interviews in op de wensen en behoeften van de ontwerpers met betrekking tot het functioneel ontwerp en wat de ontwerpers hindert in hun taken rond het functioneel ontwerp.

4. Praktijkonderzoek: op basis van analyse van de standaarden en richtlijnen, stel ik een model op van functioneel ontwerpdocumentatie. Het functioneel ontwerpmodel in hoofdstuk 3 beschrijft daarmee de huidige situatie van de functioneel ontwerpdocumentatie.

Door analyse van deze onderzoeksgegevens kom ik tot een lijst van eisen waaraan functioneel ontwerpdocumentatie dient te voldoen. Dit is de basis voor een voorstel om de huidige functionele ontwerpdocumentatie te verbeteren.

3. Functioneel ontwerp bij Quinity

Dit hoofdstuk beschrijft functioneel ontwerp bij Quinity zoals dat nu wordt gedocumenteerd. Het moet precies duidelijk zijn welke onderdelen er in het functioneel ontwerp van Quinity worden beschreven en hoe die onderdelen met elkaar samenhangen. Op basis van het praktijkonderzoek en het functioneel ontwerpmodel is dan aan te wijzen op welke plaats in het functioneel ontwerp verbeteringen aangebracht kunnen worden (qua documentatiestructuur). Met het oog op het voorstel in hoofdstuk 6 is het functioneel ontwerpmodel beschreven in UML. Het model is gebaseerd op de 'Functioneel Ontwerp standaarden en richtlijnen' (Guitink en Bron, 2008) van Quinity. Dit document beschrijft de producten van het functioneel ontwerp. Dat functioneel ontwerp bestaat uit vijf soorten producten. Eén van die producten is de functionele beschrijving. Dit is het product waar de meeste tijd aan wordt besteed in elk project. In de standaarden en richtlijnen is beschreven welke hoofdstukken en paragrafen er in moeten staan en wat daarin moet worden beschreven. Zo moet er in een functionele beschrijving bijvoorbeeld altijd een hoofdstuk staan met het te gebruiken vocabulaire; de 'concepten' van het te bouwen systeem.

Het functioneel ontwerpmodel is een manier om te documenteren wat er in een functioneel ontwerp hoort te staan. Je maakt met het model expliciet wat nu in de standaarden en richtlijnen verwoord staat en wat impliciet als kennis bij de Quinity-medewerkers aanwezig is. Het model stelt Quinity in staat om op een eenvoudige manier opgedane ervaringen met functioneel ontwerp te documenteren. Daarmee vormt het model dus een manier van kennismanagement. Ook kun je op basis van het model redeneren over het functioneel ontwerp. Door een model te maken, kom je misschien tot de ontdekking dat er dingen niet goed gedefinieerd zijn of dat er dingen ontbreken. Wanneer je een model hebt, kun je redeneren over de eigenschappen van wat je modelleert: "Is het handig om het zo te doen (met het oog op het doel van het model)?", "Welke onderdelen hebben met elkaar te maken?", "Missen er relevante onderdelen?".

Als laatste dient het UML-model als basis voor eventuele ondersteuning van functioneel ontwerpdocumentatie met geautomatiseerde tools. Wanneer je dat wilt doen, is het nodig dat je weet wat die tools moeten vastleggen. De tools 'begrijpen' de dingen die je in het functioneel ontwerp invoert. Uiteraard is dat 'begrijpen' maar ten dele, alleen wat je vastlegt in het model kan de tool 'begrijpen'.

Paragraaf 3.1 maakt eerst duidelijk wat een model is en waarvoor een model te gebruiken is. Daarna volgt in paragraaf 3.2 een korte introductie in UML om de lezer die daarmee onbekend is wegwijs te maken in UML. In paragraaf 3.3 wordt het functioneel ontwerpmodel stap voor stap opgebouwd. De eerste subparagraaf beschrijft de verschillende producten van functioneel ontwerp en de andere subparagrafen werken ieder een deel van die producten uit. Het hoofdstuk sluit af met een overzicht van het functioneel ontwerpmodel.

3.1. Wat is een model?

(Proper et al., 2005) beschrijft een model als "*a purposely abstracted and unambiguous conception of a domain.*". Dat betekent dus dat een model een *abstractie* van een bepaald domein is. Deze abstractie is een ondubbelzinnige weergave van dat domein. Het model stel je op met het oog op een bepaald doel ("*a **purposely** abstracted (...)*"). Ook DSDM geeft een definitie voor een model (DSDM-Consortium, 2008): "*Model: an abstraction of some characteristic of the business or system, as seen from a particular viewpoint.*". Ook in deze definitie komt terug dat een model een abstractie is van een bepaald domein (in dit geval de business of het systeem). Daarnaast geeft deze definitie ook aan dat een model een bepaalde kijk geeft op hetgene wat je modelleert: ("*(...) as seen from a particular viewpoint.*").

Een landkaart is een voorbeeld van een model. Het is een abstractie van de werkelijkheid, het beschrijft maar bepaalde aspecten van de werkelijkheid. Een landkaart is bedoeld om een route naar een bepaalde plaats te vinden. Op de landkaart staan dan alleen de wegen. Een ander voorbeeld is dat de landkaart alleen soorten grondgebruik aangeeft. Deze heeft dan bijvoorbeeld als doel bouwactiviteiten te kunnen plannen. Ieder model beschrijft dus de relevante eigenschappen van iets dat je modelleert met het oog op een bepaald doel.

De makers van UML beschrijven vier redenen om te modelleren (Booch et al., 2005):

1. Modellering helpt om een systeem te visualiseren zoals het is of zoals het zou moeten zijn;
2. Modellering maakt het mogelijk om de structuur en het gedrag van een systeem te specificeren;
3. Modellering geeft een sjabloon als leidraad voor de bouw van een systeem;
4. Modellering documenteert de beslissingen rond het ontwerp.

Deze redenen sluiten ook aan bij de redenen voor het functioneel ontwerpmodel dat beschreven is in paragraaf 3.3. Eerst volgt een korte introductie in UML om de lezer die daar onbekend mee is wegwijs in te maken.

3.2. Introductie UML

Om de lezer die onbekend is met UML wegwijs te maken, staat hier in het kort een overzicht van de verschillende onderdelen van deze taal. Voor verdere uitleg over deze taal verwijs ik naar (Booch et al., 2005) en (Rumbaugh et al., 2004). De meest recente versie van de UML-standaard is te vinden op `http://www.uml.org/`.

De UML bevat drie soorten bouwstenen (naar Booch et al. (2005)):

1. Things (Dingen)
2. Relationships (Relaties)
3. Diagrams (Diagrammen)

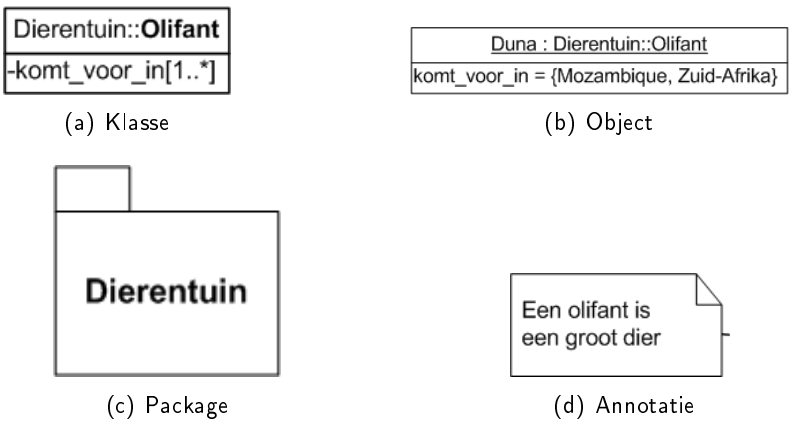
Hiermee kun je de relaties tussen dingen weergeven in verschillende soorten diagrammen die ieder een bepaalde invalshoek beschouwen. Binnen deze verschillende bouwstenen zijn verdere onderverdelingen te maken. Ik geef alleen verdere uitleg van die dingen die ik ook in het functioneel ontwerpmodel gebruik.

Er is een aantal soorten 'Dingen', waarvan ik de volgende drie gebruik:

1. Structural things (Structuur-dingen)
2. Grouping things (Groepering-dingen)
3. Annotational things (Annotatie-dingen)

De klasse is een structuur-ding. In figuur 3.1(a) is een klasse weergegeven. Met een klasse (class) kun je dingen representeren die dezelfde eigenschappen, relaties en betekenis delen. De klasse in deze illustratie heeft de naam 'Olifant'. De klasse Olifant is gegroepeerd in de package 'Dierentuin', dit is weergeven door de naam van de package voor de naam van de klasse te zetten, gescheiden door '::'. Een klasse kan één of meer attributen bezitten. Hier is dat bijvoorbeeld het attribuut 'komt_voor_in'. Met '[1..*]' wordt de *multipliciteit* van het attribuut aangegeven. Dit is het aantal toegestane waarden voor objecten met dit attribuut. Hier is weergegeven dat ieder Olifant-object minimaal één waarde heeft voor het attribuut 'komt_voor_in'. In het algemeen geeft de notatie 'i..j' met i de minimumwaarde aan en met j de maximumwaarde. Daarbij staat een * voor 'veel'. In plaats van een interval (zoals 'i..j') kan er ook één getal staan, zeg j. Dat betekent dan dat ieder object precies j waarden heeft voor dat attribuut. Hier kan ook een * gebruikt worden om 'veel' aan te geven.

Een object is ook een structuur-ding. Een object geeft een instantie van een klasse weer. De naam en het type van het object zijn onderstreept, zoals bij 'Duna' in figuur 3.1(b). Het type van het object is gescheiden van de naam door een ':'. In dit geval is het object 'Duna' van het type 'Olifant', uit de package 'Dierentuin'. Een package is een groepering-ding, een voorbeeld staat in figuur 3.1(c). De packages zijn er om dingen te groeperen. Een annotatie is een annotatie-ding, een voorbeeld staat in figuur 3.1(d). Met annotaties kun je een tekstuele toelichting geven bij bepaalde onderdelen van het model.



(a) Klasse (b) Object

(c) Package (d) Annotatie

Figuur 3.1.: Dingen

Er is een aantal soorten 'Relaties', waarvan ik de volgende drie gebruik:

1. Association (Associatie)
2. Generalization (Generalisatie)
3. Dependency (Afhankelijkheid)

De relaties kunnen tussen twee structuurdingen voorkomen. Bijvoorbeeld tussen twee klassen. De Associatie-relatie (zie figuur 3.2(a)) geeft aan dat twee dingen aan elkaar gerelateerd zijn. Een verbijzondering hiervan is de 'compositie'-relatie (zie figuur 3.2(b)). Deze geeft een deel-geheel relatie aan. De verschillende delen bestaan niet los van het geheel. De generalisatie-relatie (zie figuur 3.2(c)) geeft aan dat een ding meer generiek ('supertype') is dan een ander ding ('subtype'). Het subtype 'erft' alle eigenschappen van zijn supertype en kan daarnaast nog andere eigenschappen bezitten. De Afhankelijkheids-relatie (zie figuur 3.2(d)) geeft aan dat een verandering in het ene ding (aan de kant van de pijl) de betekenis van het andere ding (de staart van de pijl) kan wijzigen.

(a) Associatie (b) Compositie

(c) Subtype (d) Afhankelijkheid

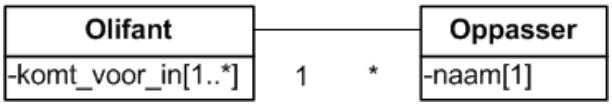
Figuur 3.2.: Relaties

Associatie-relaties (en dus ook compositie-relaties) kunnen net als klasse-attributen een multipliciteit hebben. Die multipliciteit is dan aan beiden uiteinden van de relatie gespecificeerd. In het klassediagram van figuur 3.3 staat bijvoorbeeld de relatie 'Oppasser vindt Olifant de liefste'. De multipliciteiten geven aan dat iedere Oppasser precies 1 Olifant de liefste vindt. En de andere kant op: iedere Olifant kan de liefste Olifant van 0 of meer Oppassers zijn.

Er is een aantal soorten 'Diagrammen', waarvan ik de volgende twee gebruik:

1. Class diagram (Klassediagram)
2. Object diagram (Objectdiagram)

Een diagram is de grafische representatie van een verzameling model-elementen. Het diagram is een graaf die is opgebouwd uit Dingen en Relaties. Elk soort diagram geeft het systeem vanuit een bepaald perspectief weer. In figuur 3.3 staat een voorbeeld van een eenvoudig klassediagram. Het klassediagram is een weergave van een aantal klasses en relaties tussen die klasses. Dit is een statische representatie van het systeem. Het objectdiagram is ook een statische representatie van het systeem, maar hier stellen de objecten concrete instanties van klassen voor. De klasse Olifant kan bijvoorbeeld het object 'Duna', een olifant-vrouwtje uit Ouwehands dierenpark, als instantie hebben. Tussen de objecten in het diagram kunnen ook verschillende koppelingen bestaan. ('Duna' is bijvoorbeeld een vrouwtje van 'Tooth', een andere olifant uit Ouwehands dierenpark.)



Figuur 3.3.: Klassediagram

3.3. Functioneel ontwerpmodel

Deze paragraaf beschrijft het model van functioneel ontwerpdocumentatie. In andere delen van de scriptie refereer ik naar dit model met het 'functioneel ontwerpmodel'. Het model is beschreven met UML (Unified Modeling Language). Doordat het functioneel ontwerpmodel gebruik maat van deze bestaande taal, is het duidelijk wat de constructen in het model betekenen. In eerste instantie werd ORM (Object Role Modeling) en ER (Entity Relationship) modellering gebruikt. ORM was echter onbekend bij de verschillende betrokkenen van het onderzoek en ER is teveel toegespitst op database-implementatie. Daarnaast is het met UML mogelijk om niet alleen de statische (structurele) aspecten van functioneel ontwerp te beschrijven, maar ook dynamische aspecten. Dit kan bij verdere modellering van functioneel ontwerp van pas komen. Ook is er in veel tools ondersteuning voor UML, wat het eenvoudiger maakt om geschikte plaatjes te maken.

3.3.1. Producten van functioneel ontwerp

Het functioneel ontwerp kan bestaan uit vijf soorten producten. In tabel 3.1 is weergegeven wat die verschillende producten beschrijven. In ieder geval zijn altijd de eerste twee producten aanwezig: een functionele beschrijving en een datamodel. Daarnaast worden afhankelijk van het project ook één of meer van de andere producten gemaakt. In (Guitink en Bron, 2008) staan alleen gedetailleerde standaarden en richtlijnen voor de functionele beschrijving. (Guitink et al., 2008) beschrijft standaarden en richtlijnen voor het datamodel. Deze standaarden en richtlijnen beschrijven in detail wat er in de functionele beschrijving en het datamodel gedocumenteerd moet worden. Van de andere producten zijn geen gedetailleerde standaarden en richtlijnen. Wel staat in (Guitink en Bron, 2008) globaal beschreven wat er in de andere producten moet worden gedocumenteerd.

De functionele beschrijving, externe interfacebeschrijving en outputbeschrijving zijn fysieke documenten. Hierin wordt met tekst en illustraties beschreven wat Quinity voor de klant gaat realiseren en hoe die functionaliteit gaat werken. Zo nodig kunnen deze documenten worden afgedrukt. Het datamodel beschrijft de gegevens en de samenhang daartussen met een ER-diagram. Het prototype bestaat uit een aantal HTML-pagina's waarmee het te bouwen informatiesysteem kan worden gesimuleerd.

Dezelfde producten zijn ook in figuur 3.4 weergegeven. Bij ieder product is aangegeven voor welke doelgroep dat product bestemd is. Dat kan met een 'tagged value'. Dit is een UML-construct waarmee metadata in een model kan worden opgenomen. Elk product heeft één of meer auteurs en elk product heeft een versienummer. Eerst wordt het Datamodel verder

Product	Omschrijving
Functionele Beschrijving	Beschrijft de functionele werking van het informatie. Soms zijn er voor één informatiesysteem meerdere functionele beschrijvingen. Voor specifieke onderdelen, zoals rapportages, zijn er dan aparte functionele beschrijvingen die met de juiste personen in de klantorganisatie kunnen worden afgestemd.
Datamodel	Beschrijft welke gegevens er worden opgeslagen en hoe die gegevens met elkaar samenhangen.
Externe Interfacebeschrijving	Beschrijft de uit te wisselen gegevens met externe systemen en welke procedures hiervoor nodig zijn.
Prototype	Laat zien hoe een gebruiker met het informatiesysteem kan interacteren. Een gebruiker kan door de schermen en de mogelijke overgangen bladeren om een indruk te krijgen hoe de functies van het informatiesysteem gaan werken.
Outputbeschrijving	Beschrijft de documenten die een gebruiker vanuit het informatiesysteem kan genereren. Hierbij wordt ook beschreven op basis van welke gegevens die documenten moeten worden samengesteld.

Tabel 3.1.: Producten in het functioneel ontwerp

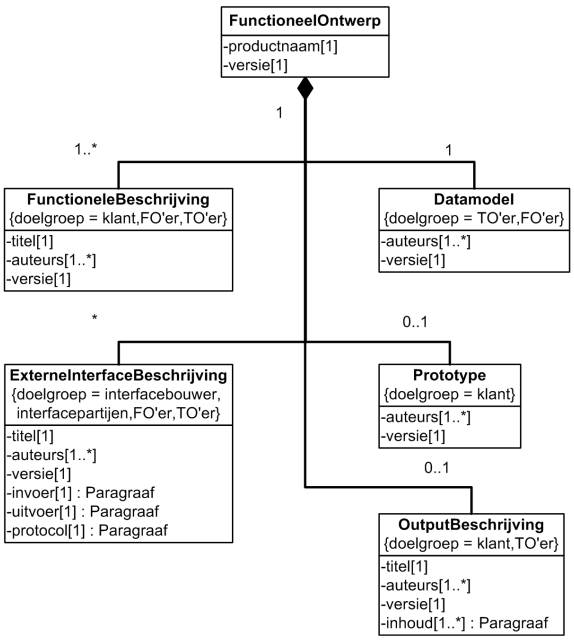
besproken, vervolgens wat er in een Paragraaf kan worden vastgelegd en welke paragraaftypen er zijn. Als laatste is beschreven hoe een FunctioneleBeschrijving is opgebouwd.

3.3.2. Het datamodel

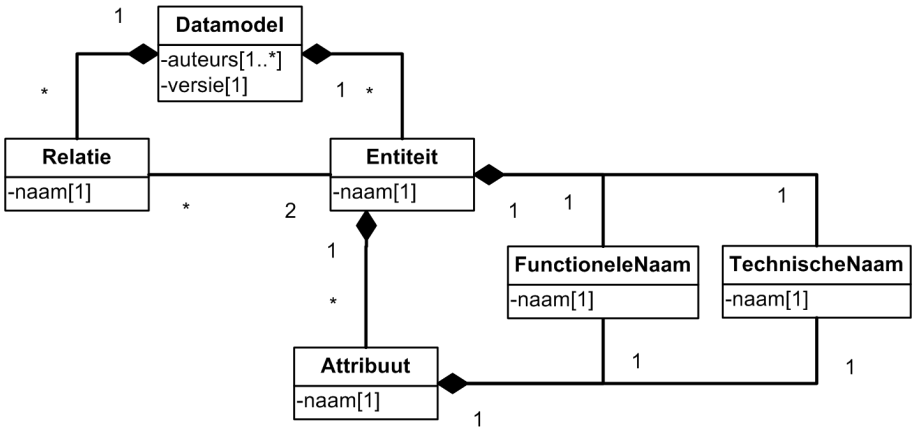
Het Datamodel representeert een ER-diagram. In figuur 3.5 is het Datamodel weergegeven. Het Datamodel bestaat uit Entiteiten en Relaties. Een Entiteit kan met 0 of meer Relaties zijn geassocieerd. Daarnaast kan een Entiteit een aantal Attributen hebben. De klassen FunctioneleNaam en TechnischeNaam zijn niet als attributen bij Entiteit en Attribuut opgenomen om te kunnen garanderen dat een FunctioneleNaam en TechnischeNaam uniek zijn in een FunctioneelOntwerp.

3.3.3. Paragraaftypes

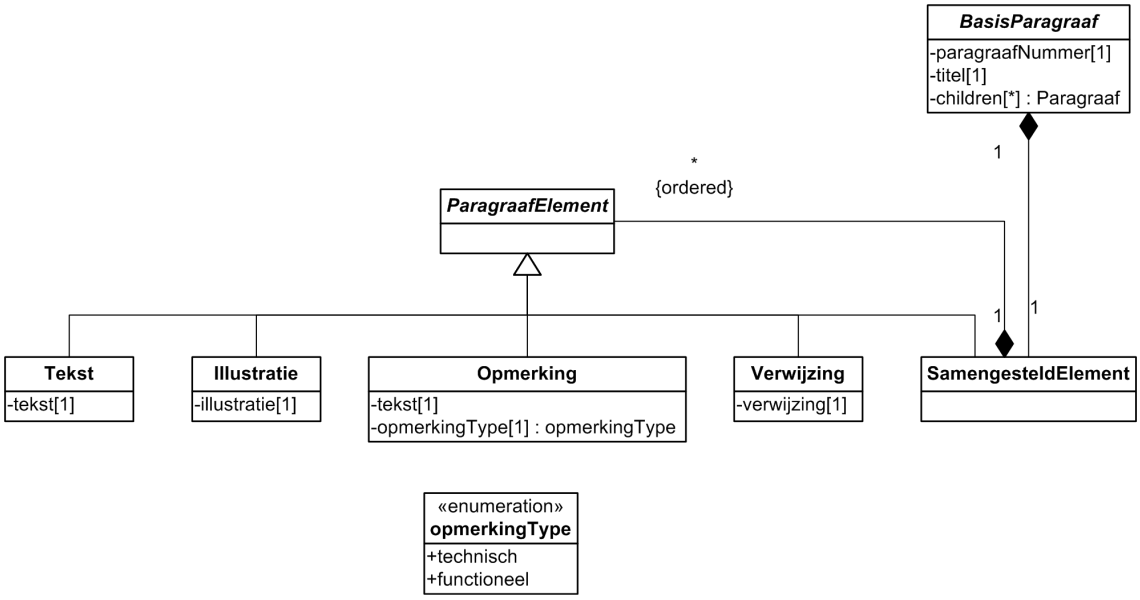
De andere producten zijn allemaal documenten die zijn opgebouwd uit paragrafen. De structuur die in alle paragrafen voorkomt, is in figuur 3.6 weergegeven. De figuur beschrijft de opbouw van de BasisParagraaf en het ParagraafElement. Er kunnen verschillende soorten paragrafen voorkomen. Dit hangt af van de verwijzingen die er mogelijk zijn vanuit de paragraaf en beschrijvingen die extra kunnen worden opgenomen in een paragraaf. Er is een abstracte superklasse BasisParagraaf, waarvan alle andere paragraaftypes de eigenschappen overnemen (zie figuur 3.6). BasisParagraaf is een abstracte klasse (weergegeven in cursief lettertype), dat wil zeggen: een BasisParagraaf komt nooit op zichzelf voor. Elk paragraaftype heeft dus



Figuur 3.4.: Producten van het functioneel ontwerp



Figuur 3.5.: Datamodel in het functioneel ontwerp



Figuur 3.6.: Paragraaftypes - de BasisParagraaf

in ieder geval een paragraafnummer en een titel. Daarnaast mogelijkerwijs een aantal onderhangende Paragrafen (de children) en één SamengesteldElement. Een SamengesteldElement kan meerdere ParagraafElementen onder zich hebben. Die ParagraafElementen staan in een bepaalde volgorde. Doordat SamengesteldElement zelf ook een ParagraafElement is, ontstaat een boomstructuur van ParagraafElementen. Een ParagraafElement kan een Tekst, Illustratie, Verwijzing, Opmerking of SamengesteldElement zijn. Hieronder staat wat ieder van de ParagraafElementen voorstelt:

- **Tekst** en **Illustratie** representeren respectievelijk tekst en illustraties.
- **Opmerking** staat voor een noot die een functioneel ontwerper bij een stukje wil maken. Er zijn twee typen opmerkingen: functioneel en technisch. Een technische opmerking is in principe alleen bedoeld voor technisch ontwerpers van Quinity.
- **Verwijzing** geeft een koppeling aan tussen twee paragrafen of tussen een paragraaf en een product. Afhankelijk van het paragraaftype waar de verwijzing gebruikt wordt, kan de verwijzing gaan naar een andere paragraaf in hetzelfde product, naar een paragraaf in een ander product, of naar een product als geheel. Een Verwijzing hoeft dus niet te verwijzen naar een paragraaf binnen hetzelfde product. Het kan ook naar andere producten verwijzen. Zo ontstaat samenhang tussen de verschillende producten van functioneel ontwerp. Op dit moment worden verwijzingen niet gestructureerd bijgehouden.
- **SamengesteldElement** is een kunstmatig ParagraafElement om de boomstructuur te kunnen maken.

De BasisParagraaf is dus de basis van alle andere paragraaftypes. In figuur 3.7 zijn alle paragraaftypes weergegeven. De afbeelding laat alleen verwijzingen tussen paragrafen zien. Paragrafen kunnen ook verwijzingen bevatten naar producten. Dit is verderop beschreven bij de

bespreking van de FunctioneleBeschrijving, in paragraaf 3.3.4. Nu worden eerst de verschillende paragraaftypes toegelicht. Een paragraaftype die voor het eerst geintroduceerd wordt, zal vetgedrukt worden weergeven.

Van de paragraaftypes is de eenvoudigste de **Paragraaf**. Dit is een normale paragraaf, zonder verwijzingen naar andere paragrafen of producten. De **DocumentRelatiesParagraaf** is een paragraaf die kan verwijzen naar andere document-producten (ExterneInterfaceBeschrijvingen, FunctioneleBeschrijvingen of OutputBeschrijvingen), zie ook figuur 3.9.

De **InterfaceParagraaf**, **SysteemOnderdeelParagraaf**, **GebruikerParagraaf** zijn paragrafen die onderdelen van een informatiesysteem representeren. In de InterfaceParagraaf wordt verwezen naar twee SysteemOnderdeelParagrafen. Dit zijn de systeemonderdelen die bij die interface betrokken zijn. Er kunnen ook gebruikers bij een systeemonderdeel betrokken zijn. Dit is gerepresenteerd door de relatie tussen SysteemOnderdeelParagraaf en GebruikerParagraaf.

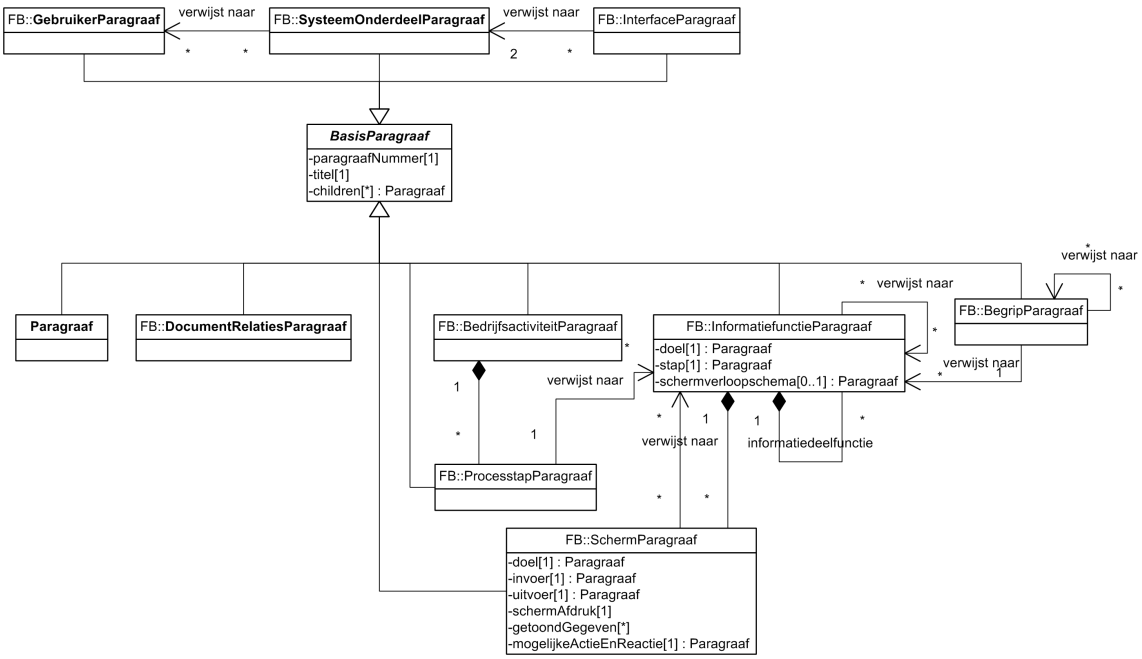
De **InformatiefunctieParagraaf** en de **SchermParagraaf** representeren de paragrafen die over een informatiefunctie en een scherm gaan. Een informatiefunctie bevat paragrafen die het doel, de stappen en het schermverloop beschrijven. Een interactieve informatiefunctie kan uit één of meerdere schermen bestaan. Die schermen worden dan beschreven in aparte SchermParagrafen. In die SchermParagrafen staat:

- een beschrijving van het doel van het scherm,
- de vereiste invoer om het scherm weer te kunnen geven (bijvoorbeeld. authenticatiegegevens),
- een schermafbeelding (een illustratie),
- een opsomming van de gegevens die getoond worden en
- een beschrijving van de mogelijke acties (en bijbehorende reacties) op het scherm.

De paragraaf over mogelijke acties en reacties kan verwijzen naar een informatiefunctie die uitgevoerd wordt wanneer een gebruiker een bepaalde actie uitvoert. Een informatiefunctie kan ook naar andere informatiefuncties verwijzen om aan te geven dat een bepaalde informatiefunctie andere informatiefuncties nodig heeft. Die ander informatiefuncties zijn dan dus informatie*deel*functies. De verwijzingen naar InformatiefunctieParagrafen kunnen ook naar andere FunctioneleBeschrijvingen binnen of buiten hetzelfde FunctioneelOntwerp wijzen.

In een **BedrijfsactiviteitParagraaf** worden één of meerdere **ProcessstapParagrafen** opgenomen. Dit geeft de stappen weer waaruit een bedrijfsactiviteit kan bestaan. Per processtap kan een verwijzing opgenomen worden naar een InformatiefunctieParagraaf die de informatiefunctie beschrijft die een gebruiker in die stap kan gebruiken. De verwijzingen naar InformatiefunctieParagrafen kunnen ook naar andere FunctioneleBeschrijvingen binnen of buiten hetzelfde FunctioneelOntwerp wijzen.

De **BegripParagraaf** beschrijft een bepaald begrip dat in het functioneel ontwerp voorkomt. Om het begrip uit te leggen, kan de BegripParagraaf verwijzen naar andere BegripParagrafen. Alle BegripParagrafen samen vormen samen een tekstuele uitleg van het datamodel. In een BegripParagraaf kunnen verwijzingen opgenomen worden naar InformatiefunctieParagrafen.



Figuur 3.7.: Overzicht van paragraaftypes

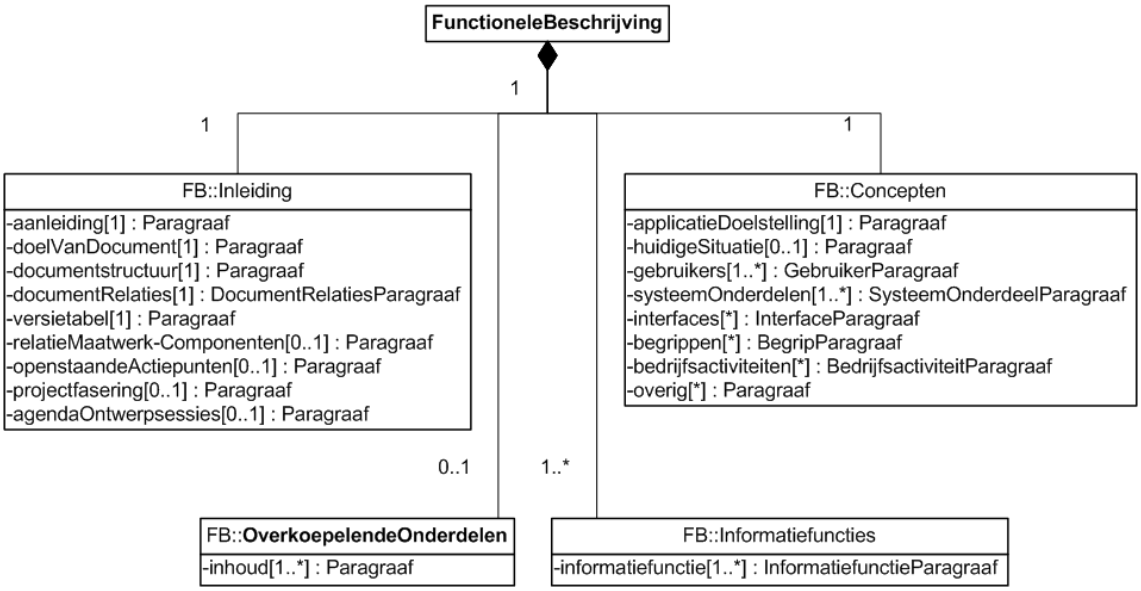
Daarmee wordt dan aangegeven hoe een bepaald begrip kan worden gebruikt door een informatiefunctie. De verwijzingen naar BegripParagrafen en InformatiefunctieParagrafen kunnen ook naar andere FunctioneleBeschrijvingen binnen of buiten hetzelfde FunctioneelOntwerp wijzen.

De paragraaftypes zijn weergegeven in figuur 3.7. Daarin zijn uitsluitend de verwijzingen opgenomen die tussen paragrafen kunnen voorkomen. Deze verwijzingen kunnen ook naar andere FunctioneleBeschrijvingen binnen of buiten hetzelfde FunctioneelOntwerp wijzen.

3.3.4. De functionele beschrijving

Nu duidelijk is welke paragraaftypes er allemaal kunnen zijn, is het mogelijk het product FunctioneleBeschrijving toe te lichten. In een functionele beschrijving beschrijft een functioneel ontwerper de functionele werking van een informatiesysteem. Een FunctioneleBeschrijving bestaat uit een aantal hoofdstukken. Dit is weergegeven door de klassen Inleiding, Concepten, OverkoepelendeOnderdelen en Informatiefuncties (in die volgorde). Deze hoofdstukken zijn in figuur 3.8 weergegeven.

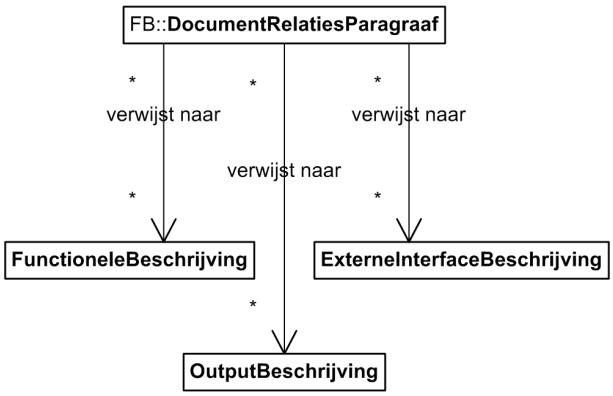
Op de documentRelaties-paragraaf na staan er in het hoofdstuk Inleiding allemaal Paragrafen. De verplichte paragrafen zijn: aanleiding, doelVanDocument, documentstructuur, documentRelaties en versietabel. Daarnaast kunnen er nog één van de paragrafen relatieMaatwerk-Componenten, openstaandeActiepunten, projectfasering en agendaOntwerpsessies voorkomen. De laatste drie paragrafen komen alleen voor als de FunctioneleBeschrijving nog niet afgerond is. De paragraaf relatieMaatwerk-Componenten komt alleen voor als er gebruik wordt gemaakt van QIS. De aanleiding-paragraaf beschrijft de aanleiding van het project. De doelVanDocument-paragraaf beschrijft waarom dit document wordt opgesteld. De documentStructuur-paragraaf



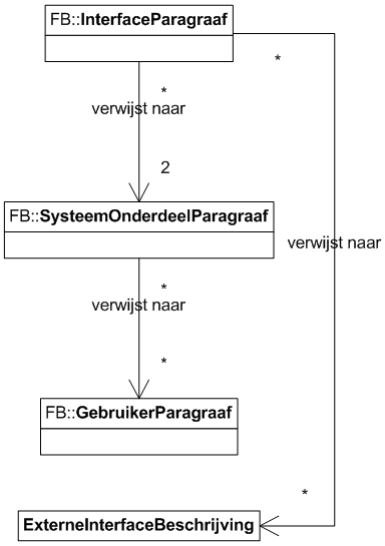
Figuur 3.8.: Hoofdstukken in een functionele beschrijving

geeft per hoofdstuk in dit document een korte samenvatting (één of twee regels). De versietabel somt de versies op die er van dit document reeds zijn geweest. De DocumentRelatiesParagraaf is een paragraaf die kan verwijzen naar andere document-producten (ExterneInterfaceBeschrijvingen, FunctioneleBeschrijvingen, QISConfiguratie of OutputBeschrijvingen), mogelijkerwijs ook van andere functionele ontwerpen. Dit is weergegeven in figuur 3.9. De paragraaf relatieMaatwerk-Componenten beschrijft hoe het maatwerk aansluit op de te gebruiken componenten. Voor ontwerpsessies kan het voor de klantmedewerkers fijn zijn om alles in één document te hebben. Hierom kan ook een lijst met openstaande actiepunten, de projectfasering en de agenda voor de ontwerpsessies in de functionele beschrijving worden opgenomen.

Na de Inleiding komt het hoofdstuk Concepten. Dit hoofdstuk beschrijft de achtergronden van het informatiesysteem, het helpt een lezer van de functionele beschrijving te begrijpen waar het



Figuur 3.9.: Documentrelaties vanuit een functionele beschrijving

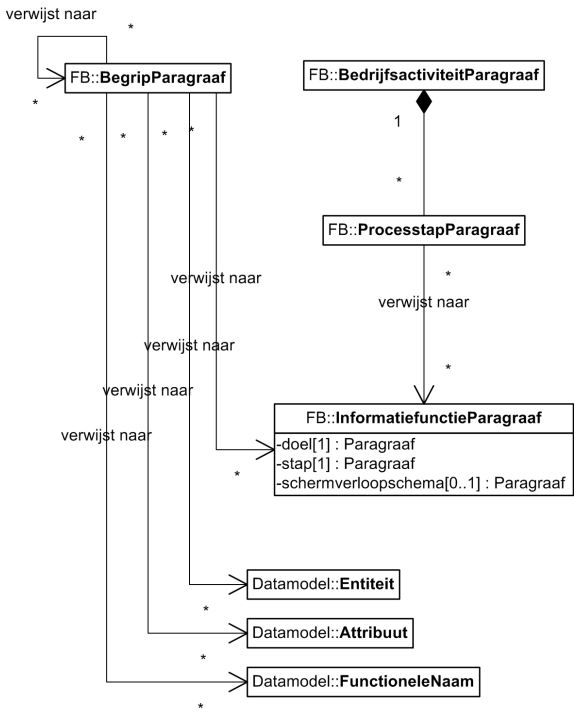


Figuur 3.10.: Interfaces, systeemonderdelen en gebruikers

over gaat. In het concepten-hoofdstuk komen de volgende paragrafen voor: applicatieDoelstelling, huidigeSituatie, gebruikers, systeemOnderdelen, interfaces, begrippen, bedrijfsactiviteiten en overige. De paragraaf applicatieDoelstelling beschrijft waarom de applicatie wordt gemaakt; welke problemen lost het op of welke vraag wordt er door beantwoord. Voor het begrijpen van het informatiesysteem kan het nodig zijn te weten op welke manier de klant nu zijn werk doet. Dit wordt dan beschreven in de paragraaf huidigeSituatie.

In figuur 3.10 zijn de relaties tussen GebruikerParagraaf, SysteemOnderdeelParagraaf, InterfaceParagraaf en ExterneInterfaceBeschrijving weergegeven. De gebruikers-paragrafen beschrijven de verschillende typen gebruikers van het informatiesysteem. De systeemonderdelen-paragrafen beschrijven de systemen waarmee het te bouwen informatiesysteem gekoppeld is en de subsystemen waaruit het informatiesysteem is opgebouwd. Daarnaast beschrijven deze paragrafen met welke systemen en subsystemen de gebruikers (uit de gebruikersparagrafen) interacteren. In de interfacesparagrafen worden de koppelingen (=interfaces) tussen systemen en subsystemen opgesomd. Per koppeling wordt in één of twee zinnen beschreven wat de koppeling omhelst. Een InterfaceParagraaf verwijst naar een ExterneInterfaceBeschrijving voor de gedetailleerde beschrijving van die interface.

In figuur 3.11 zijn de relaties tussen BegripParagraaf, BedrijfsactiviteitParagraaf, ProcesstapParagraaf, InformatiefunctieParagraaf en andere producten weergegeven. In de begrippenparagrafen wordt per begrip in een BegripParagraaf telkens een begrip uitgelegd. De uitleg van deze begrippen is in feite een tekstuele uitleg van het Datamodel. Vanuit een BegripParagraaf kan dan ook verwezen worden naar Attribuut, Entiteit en FunctioneleNaam uit het Datamodel. Daarnaast kan een BegripParagraaf verwijzingen naar andere BegripParagrafen bevatten als voor de uitleg van een begrip een ander begrip nodig is. Het is de bedoeling dat eerst de begrippen worden uitgelegd die zonder gebruik van andere begrippen uitgelegd kunnen worden. In een BegripParagraaf kunnen verwijzingen opgenomen worden naar InformatiefunctieParagrafen. Daarmee wordt dan aangegeven hoe een bepaald begrip kan worden gebruikt door een informa-



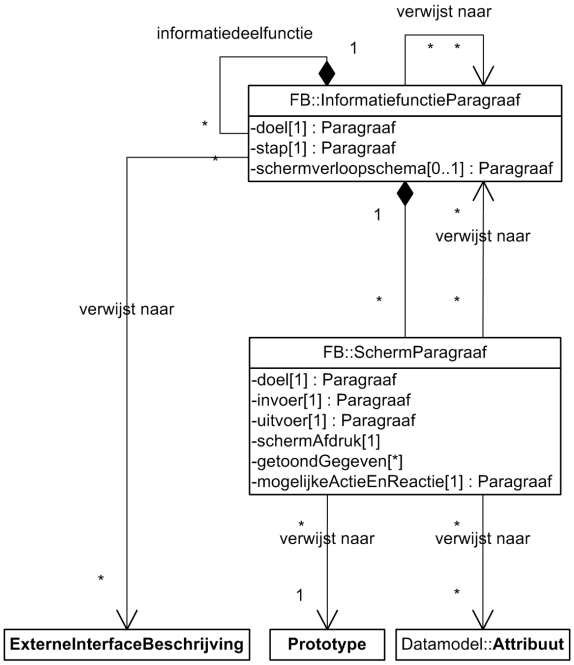
Figuur 3.11.: Begrippen en bedrijfsactiviteiten

tiefunctie. De verwijzingen naar BegripParagrafen en InformatiefunctieParagrafen kunnen ook naar andere FunctioneleBeschrijvingen binnen of buiten hetzelfde FunctioneelOntwerp wijzen.

Na de BegripParagrafen in het Concepten-hoofdstuk komen de BedrijfsactiviteitParagrafen. Dit paragraaftype is eerder al besproken. Omdat er misschien nog andere onderwerpen aan bod moeten komen, kan een ontwerper in de overige-paragrafen die andere onderwerpen beschrijven. Dit zijn normale Paragrafen.

Het derde hoofdstuk dat aan bod komt is het hoofdstuk 'OverkoepelendeOnderdelen'. Hierin staan onderdelen beschreven die voor het systeem als geheel gelden. Daarbij valt te denken aan grafische vormgeving en autorisatie. Dit wordt beschreven in één of meer Paragrafen.

Als laatste zijn er één of meer hoofdstukken over Informatiefuncties. Deze beschrijven de informatiefuncties van het informatiesysteem. In één hoofdstuk kunnen beschrijvingen staan van meerdere informatiefuncties. Vaak is het dan één 'hoofd'-informatiefunctie die meerdere informatiedeelfuncties onder zich heeft. Een informatiefunctie 'beheren polissen' kan bijvoorbeeld uiteenvallen in meerdere informatiefuncties die ieder een deel van het beheer beschrijven. Dit zijn dan informatiedeelfuncties. Het is aan de ontwerper om een logische indeling te maken. In de beschrijving van de paragraaftypes is aangegeven hoe een InformatiefunctieParagraaf en SchermParagraaf kan verwijzen naar één of meer InformatiefunctieParagrafen. Daarnaast kan een InformatiefunctieParagraaf verwijzen naar een ExterneInterfaceBeschrijving, een SchermParagraaf verwijst naar het Prototype en kan verwijzen naar Attributen uit het Datamodel. De relaties tussen InformatiefunctieParagraaf, Schermparagraaf en andere (onderdelen van) producten zijn weergegeven in figuur 3.12.



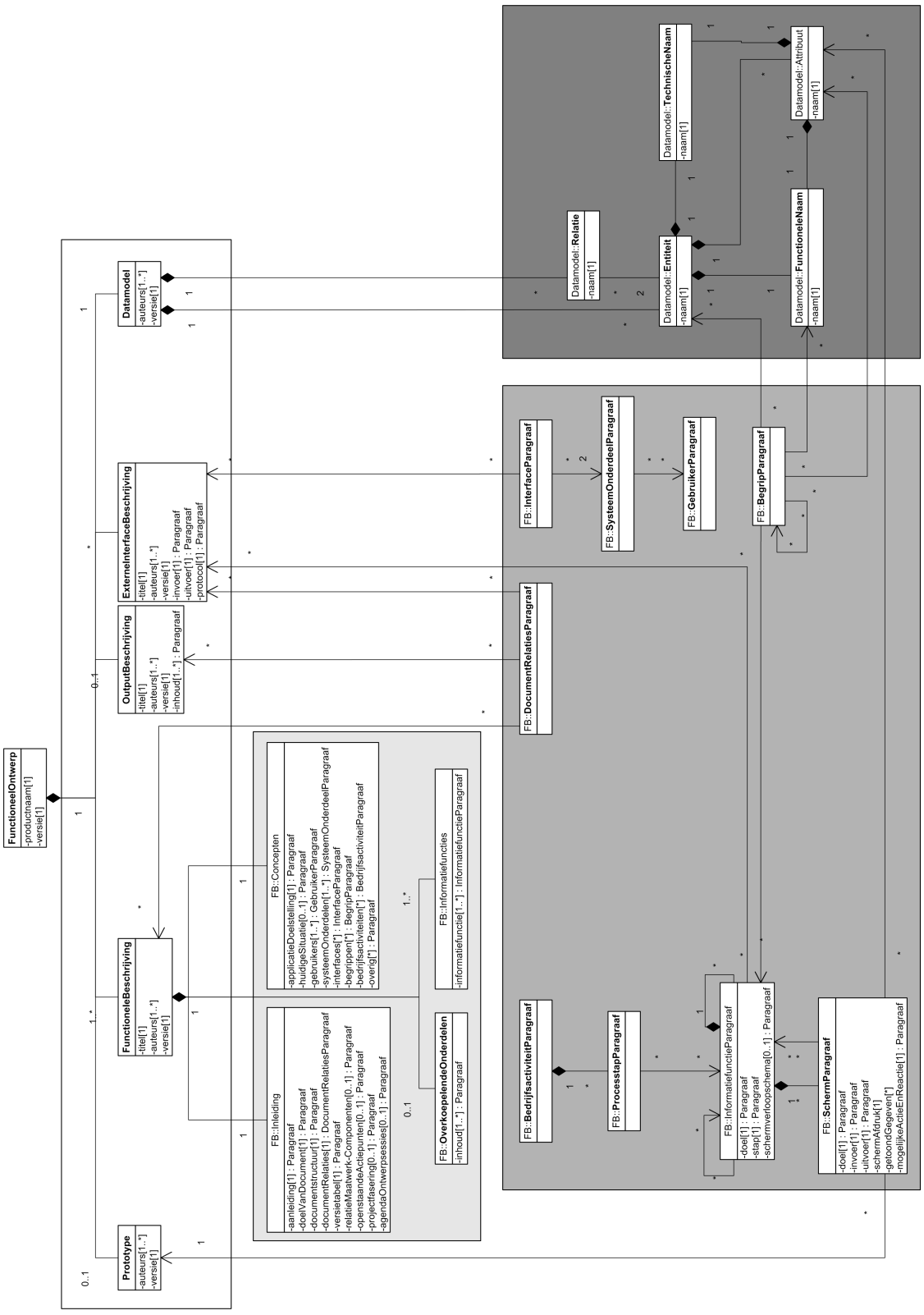
Figuur 3.12.: Informatiefuncties

3.3.5. Overzicht van het functioneel ontwerpmodel

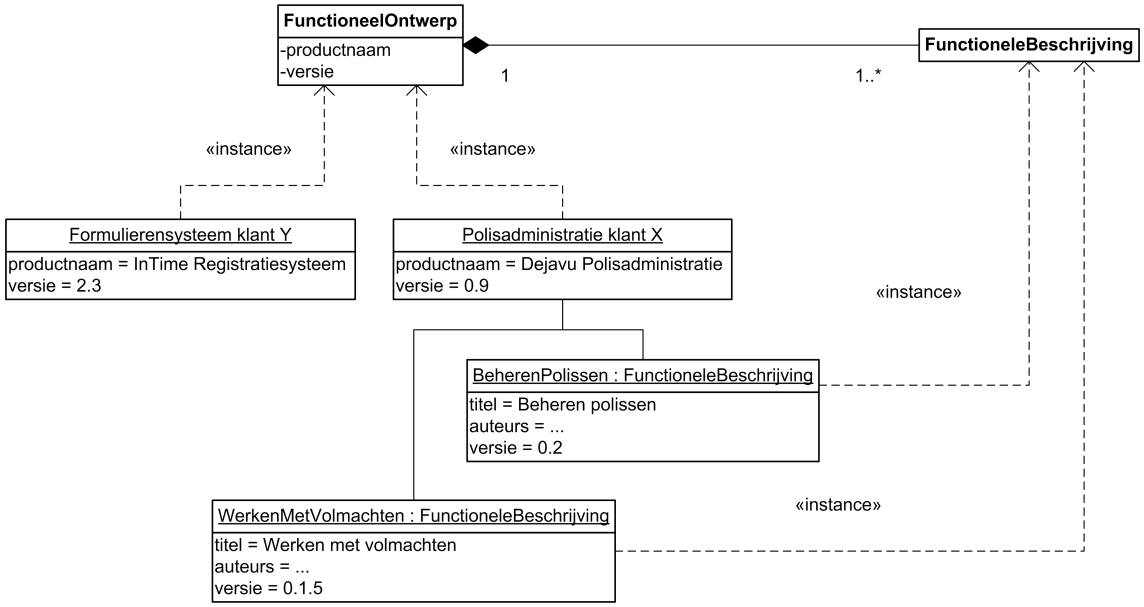
In figuur 3.13 staat een overzicht van het functioneel ontwerpmodel als geheel van producten, paragrafen en verwijzingen daartussen. Het gedeelte met het witte kader bevat de producten van functioneel ontwerp. In het lichtgrijze kader staan de hoofdstukken van een functionele beschrijving. Het iets donkerdere grijze kader bevat de verschillende paragraaftypes die in een functioneel ontwerp mogelijk zijn (op de standaard Paragraaf na). Het donkergrijze kader bevat de onderdelen van het datamodel. De gerichte associaties (de pijlen) geven verwijzingen tussen de verschillende onderdelen van functioneel ontwerp aan.

Wanneer functioneel ontwerpen worden opgeslagen volgens dit functioneel ontwerpmodel, kun je die functionele ontwerpen zien als instanties van het functioneel ontwerpmodel. Dit is weergegeven in figuur 3.14. Hier zijn twee instanties van FunctioneelOntwerp weergegeven: 'Formulierensysteem klant Y' en 'Polisadministratie klant X'. Tevens zijn er in de figuur twee instanties van FunctioneleBeschrijving weergegeven: BeherenPolissen en WerkenMetVolmachten. Deze FunctioneleBeschrijvingen horen bij het FunctioneelOntwerp 'Polisadministratie klant X'. De inhoud van de FunctioneleBeschrijvingen is verder niet weergegeven in de figuur

Hiermee is het model van functioneel ontwerp bij Quinity beschreven. Het functioneel ontwerp bestaat uit een aantal verschillende producten. De functionele beschrijving, externe interfacebeschrijving en de outputbeschrijving zijn documentproducten. Daarnaast zijn er het datamodel en het prototype. Binnen de producten en tussen de producten kunnen verwijzingen voorkomen. Het volgende hoofdstuk zal in kaart brengen wat de wensen en behoeften zijn van de betrokkenen bij deze ontwerpdocumentatie. Op basis van die ervaringen zal ik eisen formuleren waaraan de functioneel ontwerpdocumentatie dient te voldoen. In het hoofdstuk over het



Figuur 3.13.: Overzicht functioneel ontwerpmodel



Figuur 3.14.: Concrete functionele ontwerpen zijn instanties van het functioneel ontwerpmodel

voorstel voor vernieuwde functioneel ontwerpdocumentatie gebruik ik deze eisen samen met het functioneel ontwerpmodel en ideeën uit andere systeemontwikkelingsmethoden om te laten zien welke verbeteringen in de functioneel ontwerpdocumentatie te maken zijn.

4. Praktijkonderzoek

Het onderzoek naar de wensen en behoeften van de betrokkenen met het functioneel ontwerp is een kwalitatief onderzoek. Hierbij is gebruik gemaakt van participerende observatie, topic-interviews en documenten (Baarda et al., 2005). In tabel 4.1 staat een overzicht van de gebruikte bronnen. Op basis van deze bronnen is nagegaan wat functioneel ontwerp bij Quinity inhoudt en wat de wensen en behoeften van de verschillende betrokkenen met betrekking tot het functioneel ontwerp zijn.

Nadat een eerste versie van het functioneel ontwerpmodel (zie paragraaf 3.3) was gemaakt, is dit model in een aantal reviewsessies aangevuld met opmerkingen van de Quinity-begeleiders om aan te sluiten bij wat er in de praktijk in het functioneel ontwerp wordt gedocumenteerd. Daarnaast is gebruik gemaakt van vijf functionele beschrijvingen om te controleren of er in functionele beschrijvingen onderwerpen worden beschreven die niet of op een andere manier in de standaarden en richtlijnen staan.

Naast dit documentonderzoek zijn er aantekingen gemaakt van drie functioneel ontwerpsessies en gesprekken met de klantmedewerkers bij die sessies. Onder de klantmedewerkers van dat project is ook een vragenlijst over functioneel ontwerp verspreid. Hierop zijn drie reacties gekomen. Omdat de ingevulde vragenlijsten slechts over één project gaan, worden de resultaten hiervan niet verder uitgewerkt en geanalyseerd. Daardoor zou een te eenzijdig beeld ontstaan van wat de wensen en behoeften van klantmedewerkers over het functioneel ontwerp zijn.

In de drie groepsinterviews met functioneel en technisch ontwerpers van Quinity is specifiek ingegaan op een aantal onderwerpen rond functioneel ontwerp. In de rest van dit hoofdstuk is de aanpak van die interviews beschreven. De resultaten en analyses van de interviews zijn vanwege de vertrouwelijkheid ervan opgenomen in een vertrouwelijke bijlage. Op basis van de resultaten, analyses, en het literatuuronderzoek is een aantal eisen opgesteld waaraan functioneel ontwerpdocumentatie dient te voldoen. Deze eisen zijn wel in dit hoofdstuk opgenomen.

4.1. Interviews met functioneel en technisch ontwerpers

Om na te gaan hoe functioneel en technisch ontwerpers het functioneel ontwerp opstellen en wat hun wensen en behoeften in hun dagelijkse werk zijn met betrekking tot het functioneel ontwerp, zijn een aantal interviewsessies gehouden. In deze paragraaf wordt nader beschreven wat het doel is van die interviewsessies en hoe de interviewsessies zijn aangepakt.

4.1.1. Doel

Het doel van de interviews is erachter te komen hoe functioneel en technisch ontwerpers werken met het functioneel ontwerp bij Quinity. Daarbij lag de focus op de relatie tussen theorie en

Soort bron	Omschrijving bron
Interne documenten	Standaarden en richtlijnen functioneel ontwerp
	Overzicht projectvoering
	Vijf functionele beschrijvingen
Gesprekken	Klantmedewerkers van één project (tijdens / na FO-sessies)
	Quinitymedewerkers (bij de lunch / koffie / bedrijfsuitjes)
	Quinitybegeleiders (tijdens tweewekelijkse overlegsessies)
Interviews	Twee groepsinterviews met dezelfde onderwerpen, één bij functioneel ontwerpers en één bij technisch ontwerpers
	Eén groepsinterview over de uitwerking en analyse van de andere twee interviews (zijn de gevonden knelpunten herkenbaar?)
	Drie reacties van klantmedewerkers uit één bepaald project op de vragenlijst over functioneel ontwerp

Tabel 4.1.: Gebruikte bronnen voor het praktijkonderzoek

praktijk: wat zijn goede punten aan de standaarden en richtlijnen en wat zijn minder goede of slechte punten. Daarnaast moeten de interviews een indicatie geven van de knelpunten die functioneel en technisch ontwerpers ervaren rond het functioneel ontwerp. De onderwerpen die aan de orde kwamen zijn:

- **Flexibiliteit**: kun je uitdrukken wat je wilt in het functioneel ontwerp binnen de kaders van de standaarden en richtlijnen;
- **Structuur**: is de indeling volgens de standaarden en richtlijnen logisch;
- **Volledigheid**: geven de standaarden en richtlijnen een volledig overzicht van wat er in een functioneel ontwerp hoort te staan;
- **Nuttigheid**: in hoeverre helpen de standaarden en richtlijnen om het functioneel ontwerp op te stellen;
- **Gebruik**: wat zijn de redenen achter het al dan niet gebruiken van de standaarden en richtlijnen.

4.1.2. Aanpak interviews

Om na te gaan hoe functioneel en technisch ontwerpers het functioneel ontwerp opstellen, en aan welke aanpassingen zij op basis van hun dagelijkse werk behoefte zouden hebben, zijn twee interviewsessies gehouden, één met functioneel ontwerpers en één met technisch ontwerpers. Om conclusies te nuanceren en nader te specificeren zijn de uitwerkingen en analyses van deze twee sessies vervolgens besproken met twee functioneel ontwerpers, van wie er één betrokken was bij het opstellen van de standaarden en richtlijnen voor functioneel ontwerp.

In de derde sessie is ook het (toen nog voorlopige) model voor functioneel ontwerp besproken. Dit model was vóór die sessie opgesteld aan de hand van de standaarden en richtlijnen, bestaande functioneel ontwerpen, feedback vanuit de eerste twee interview-sessies en feedback van de

begeleiders. Het model heeft als doel de huidige situatie te beschrijven. Door de feedback in dit derde interview is gevalideerd of het model overeenkomt met de werkelijkheid (juistheid) en of het compleet is (volledigheid). Het voorstel in hoofdstuk 6 gaat uit van het model van de huidige situatie.

Voor het interview met functioneel ontwerpers is een groepje geselecteerd met de onderstaande eigenschappen. Dit om een zo volledig en gedetailleerd mogelijk beeld te krijgen van de werkwijze van functioneel ontwerpers met het functioneel ontwerp.

- Er is iemand die later in een project is ingestroomd. Deze persoon kan meepraten over hoe goed iemand van buiten een lopend project de begrippen en concepten kan begrijpen en hoe goed deze persoon daarmee kan werken.
- Er is iemand die al geruime tijd ervaring heeft met het opstellen van functionele ontwerpen.
- Alle ontwerpers hebben ervaring met functioneel ontwerp in grotere projecten.
- Er is iemand die ervaring heeft met functioneel ontwerp in projecten waar maatwerk wordt ontwikkeld, waarbij geen gebruik wordt gemaakt van de standaardcomponenten van Quinity.
- Er is iemand die ervaring heeft met functioneel ontwerp in projecten waar standaardcomponenten worden ingezet samen met een deel maatwerk-ontwikkeling.

Voor het interview met technisch ontwerpers is een groepje geselecteerd met onderstaande eigenschappen. Dit om een zo volledig en gedetailleerd mogelijk beeld te krijgen van de werkwijze van technisch ontwerpers met het functioneel ontwerp.

- Er is iemand die al geruime tijd ervaring heeft met het opstellen van technische ontwerpen aan de hand van functionele ontwerpen.
- Er is iemand die ervaring heeft met het opstellen van technische ontwerpen in projecten waar maatwerk wordt ontwikkeld, waarbij geen gebruik wordt gemaakt van de standaardcomponenten van Quinity. .
- Er is iemand die ervaring heeft met het opstellen van technische ontwerpen in projecten waar standaardcomponenten worden ingezet samen met een deel maatwerk-ontwikkeling.

De vragen van de interviews zijn aan de ene kant beschrijvend van aard (wat **is** functioneel ontwerp op dit moment) en aan de andere kant evaluerend (wat **vind je** van het functioneel ontwerp op dit moment). In bijlage C staan de vragen uit de interviews van functioneel en technisch ontwerpers.

In het derde interview zijn twee mensen geselecteerd die veel ervaring met functioneel ontwerp hebben en die nog niet in eerdere sessies input hadden gegeven voor het onderzoek.

4.2. Eisen voor vernieuwde functioneel ontwerpdocumentatie

De resultaten en analyses van de interviews zijn in het document 'Vertrouwelijke bijlagen Behorende bij de masterscriptie "Gestructureerde documentatie van functioneel ontwerp"' te vinden. Op basis van de uitwerkingen en analyses daarvan geef ik in de volgende subparagrafen een overzicht van de eisen voor de vernieuwde functioneel ontwerpdocumentatie.

4.2.1. Onderhoudbaarheid

Gedurende het bestaan van documentatie worden wijzigingen op die documentatie gedaan. Goede *onderhoudbaarheid* houdt in dat het eenvoudig en snel mogelijk moet zijn om die wijzigingen op een consistente en volledige manier door te voeren. Omdat de dingen die in de documentatie beschreven worden met elkaar samenhangen, kan het zijn dat door een wijziging de consistentie van de documentatie niet meer klopt. Documentatie is beter onderhoudbaar naarmate het minder tijd kost een wijziging op een consistente manier door te voeren. Daarnaast is documentatie beter onderhoudbaar naar mate een wijziging zo eenvoudig mogelijk kan worden doorgevoerd.

4.2.2. Traceerbaarheid

Van een informatiefunctie moet duidelijk zijn uit welke requirement(s) die informatiefunctie voortvloeit en op welke plaats(en) in het technisch ontwerp die informatiefunctie wordt uitgewerkt. Per informatiefunctie moet dus direct een overzicht gegeven kunnen worden van de bijbehorende requirements en bijbehorende paragrafen in het technisch ontwerp. Daarnaast moeten non-functionele requirements direct kunnen verwijzen naar het paragrafen in het technisch ontwerp. Er moeten dus verwijzingen aangebracht kunnen worden tussen:

- Requirements en functioneel ontwerp
- Functioneel ontwerp en technisch ontwerp
- Requirements en technisch ontwerp

4.2.3. Annotaties

Soms is het nuttig om informatie in de functioneel ontwerpdocumentatie te annoteren met aanvullende informatie die niet direct onderdeel is van het ontwerp zelf: *annotaties*. Zo kan aanvullende informatie onderscheiden worden van informatie uit het ontwerp zelf. In ieder geval moeten de volgende soorten annotaties gemaakt en onderscheiden kunnen worden:

Functionele opmerking beschrijft een opmerking over de functionaliteit. Dit is aanvullende informatie die niet direct nodig is om het functioneel ontwerp te begrijpen.

Technische opmerking beschrijft hoe een functionaliteit geïmplementeerd zou kunnen worden, of waar een ontwikkelaar rekening mee moet houden bij implementatie.

Ontwerpbeslissing beschrijft de keuze waarom een functionaliteit op een bepaalde manier is ontworpen. Mogelijk worden hier ook ontwerpalternatieven beschreven.

4.2.4. Intelligente zoekmogelijkheden

Vooral voor junior-ontwerpers en klantmedewerkers die later in een project instromen is het nuttig dat ze snel en eenvoudig in de functioneel ontwerpdocumentatie kunnen vinden wat ze zoeken. Daarvoor zijn *intelligente zoekmogelijkheden* nodig. Daarnaast worden intelligente

zoekmogelijkheden ook belangrijker naar mate de omvang van de documentatie van de standaardcomponenten toeneemt. Naast het zoeken op basis van zoekwoorden of zoekzinnen dient er in ieder geval gezocht te kunnen worden op basis van:

Type-informatie dit maakt het mogelijk dat een gebruiker alleen in bepaalde typen onderdelen van de functioneel ontwerpdocumentatie zoekt. Bijvoorbeeld alleen in documentatie van informatiefuncties, begrippen of datamodellen.

Versie-informatie dit maakt het mogelijk dat een gebruiker alleen in bepaalde versies van functioneel ontwerpdocumentatie zoekt. Bijvoorbeeld alleen de laatste drie versies van een functioneel ontwerp.

5. Functioneel ontwerp in drie hedendaagse methoden

Een functioneel ontwerp of daarmee vergelijkbaar concept komt in veel systeemontwikkelingsmethoden voor. In dit hoofdstuk wordt een aantal systeemontwikkelingsmethoden naast elkaar gezet om te onderzoeken op welke manier die methoden functioneel ontwerp documenteren. Daarnaast kunnen deze methoden interessante punten aan het licht brengen die kunnen helpen bij verbetering van het functioneel ontwerp van Quinity. De methoden in de vergelijking zijn SDM / LAD, DSDM en RUP. Dit zijn veelgebruikte systeemontwikkelingsmethoden bij IT-bedrijven en IT-afdelingen in Nederland (Neering et al., 2005). SDM en LAD zijn lineaire methoden waarbij LAD een voortzetting is van SDM. In het overzicht zijn deze twee methodes dan ook samengenomen. RUP en DSDM zijn beiden evolutionaire methoden. Door naast elkaar te zetten wat deze verschillende methoden zien als functioneel ontwerp, geef ik aan welke aspecten bij het functioneel ontwerp een rol spelen. Iedere methode heeft zijn eigen wijze van documentatie.

Eerst zijn de processen van de drie systeemontwikkelingsmethoden in vogelvlucht beschreven. Daarna volgt per methode een overzicht van de onderwerpen die in het functioneel ontwerp van die methode beschreven wordt. Niet al die onderwerpen zijn voor alle betrokkenen bij het functioneel ontwerp bestemd. Daarom is in datzelfde overzicht per methode ook aangegeven wat de verschillende doelgroepen van die onderwerpen zijn.

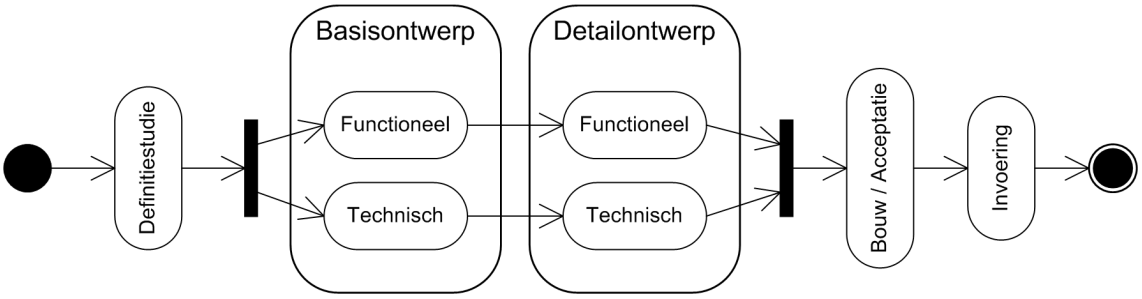
Voor het overzicht is gebruik gemaakt van de onderstaande bronnen:

- SDM / LAD: (Fokkinga et al., 2002, LAD Het lineair ontwikkelen van informatiesystemen)), (Vlasblom et al., 1991, Produkten van systeemontwikkeling)
- DSDM: (DSDM-Consortium, 2008, DSDM Manual versie 4.2)
- RUP: (Kruchten, 2003, The Rational Unified Process: an introduction)

5.1. Overzicht van drie ontwikkelingsmethoden

5.1.1. SDM / LAD: documenten

SDM en LAD zijn bedoeld voor het lineair ontwikkelen van informatiesystemen. De fasen van een project dat met SDM / LAD wordt uitgevoerd zijn achtereenvolgens Definitiestudie, Basisontwerp (globaal functioneel en technisch ontwerp), Detailontwerp (gedetailleerd functioneel en technisch ontwerp), Bouw / Acceptatietest, Invoering. De klant is vooral betrokken bij de Definitiestudie, het Functioneel Ontwerp en bij de Acceptatietest. Dit is tegelijk ook het



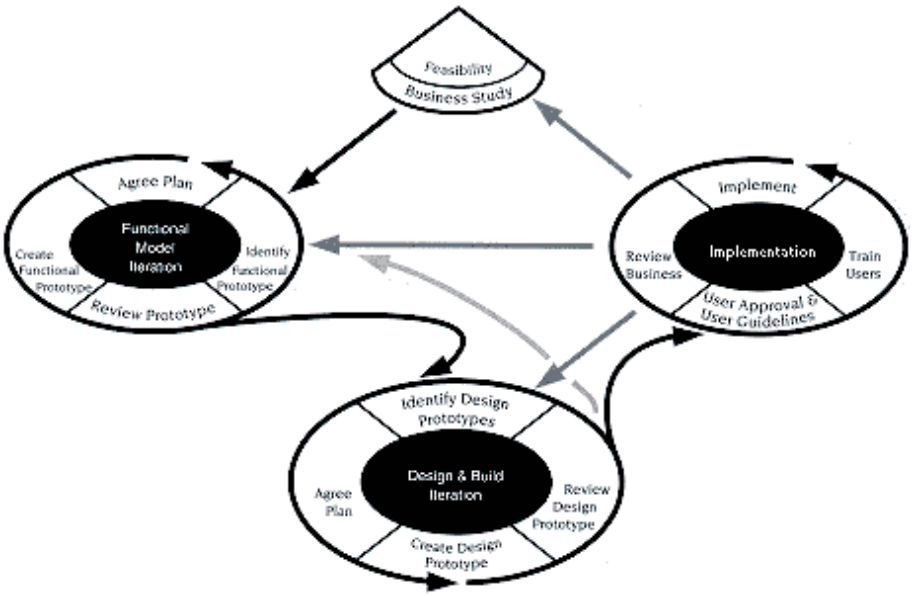
Figuur 5.1.: Overzicht van het SDM/LAD-proces

grootste probleem van deze methoden: tussen afronding van het ontwerp en afronden van de bouw zit soms een grote periode. In die periode kunnen de wensen van de klant veranderen, maar die worden niet meer opgenomen in het systeem. Ook kunnen de wensen van de klant niet goed begrepen zijn door de systeemontwikkelaars. Hierdoor is de kans aanwezig dat bij afronding van het project de klant een systeem krijgt dat niet of niet helemaal aan zijn wensen voldoet. Aan de andere kant is deze methode goed bruikbaar wanneer de wensen van de klant niet snel wijzigen, of voor kortlopende projecten. Het is dan voor elke fase duidelijk wat er moet gebeuren. Een overzicht van dit proces is in figuur 5.1 weergegeven.

5.1.2. DSDM: prototypes

DSDM is een iteratieve methode. Het biedt een raamwerk van instrumenten biedt om systemen onder strakke tijdsbeperkingen te bouwen en onderhouden. Vaak wordt DSDM tegelijk ook gebruikt als een incrementele methode. Per increment wordt er dan in een aantal iteraties een voor de klant bruikbaar product opgeleverd. Er zijn vijf fasen in DSDM te onderscheiden: Haalbaarheidsstudie, Bedrijfsstudie, Functioneel Model Iteratie, Ontwerp en Bouw Iteratie, Implementatie. De Haalbaarheidsstudie en de Bedrijfsstudie worden aan het begin van een project achter elkaar uitgevoerd. De andere drie fasen hebben ieder vier subfasen. Deze vier sequentiële subfasen vormen één iteratie. In de Functioneel Model Iteratie heb je zo bijvoorbeeld de subfasen Identificeer Functioneel Prototype, Kom Planning Overeen, Maak Functioneel Prototype en Review Prototype. Na de laatste subfase kan er worden doorgegaan naar de Ontwerp en Bouw Iteratie, maar er kan ook (al dan niet parallel) een tweede Functioneel Model Iteratie worden gestart. Verschillende iteraties in een project kunnen zich dus ieder in verschillende fasen bevinden.

Het kenmerkende van DSDM is dat er telkens op basis van een prototype wordt gewerkt. De klant wordt in elke fase van het project betrokken door commentaar te geven op het prototype. Door de visualisatie krijgen de klant en de ontwikkelaar meer begrip van het systeem dat gemaakt moet worden. Eventuele onduidelijkheden in eerdere communicatie worden zo opgevangen en verbeterd. Een overzicht van het DSDM-proces is in figuur 5.2 weergegeven.



Figuur 5.2.: Overzicht van het DSDM-proces

5.1.3. RUP: use-cases

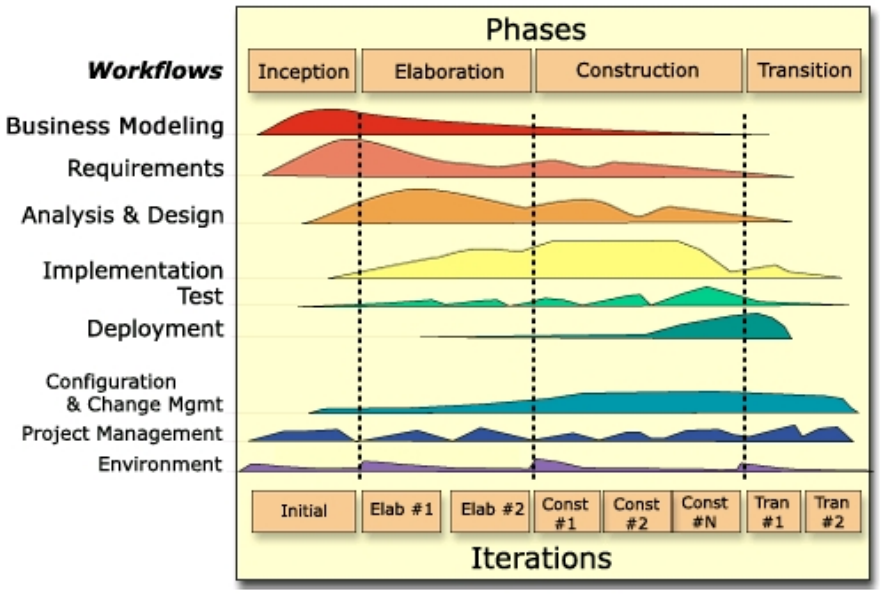
RUP is een iteratieve en incrementele methode. Er zijn vier fasen: Aanvang (Inception), Detaillering (Elaboration), Constructie en Transitie. Binnen deze fases zijn een aantal activiteiten die in iedere fase plaatsvinden. Wel legt iedere fase de nadruk op bepaalde activiteiten, zie figuur 5.3. Dit zijn Bedrijfsmodellering, Requirementsanalyse, Analyse & Ontwerp, Implementatie, Test en Deployment. Vanaf het begin wordt gebruik gemaakt van use-cases. Deze beschrijven de interacties van actoren (mensen, systemen) met het te ontwikkelen systeem. Naast deze reguliere use-cases zijn er ook business use-cases. Deze beschrijven de bedrijfsprocessen en kunnen ook over niet-geautomatiseerde processen gaan. De klant is bij iedere fase betrokken. Een overzicht van dit proces is in figuur 5.3 weergegeven.

5.2. Het functioneel ontwerp in drie methoden

Ieder van de drie methoden geeft aan wat er gedocumenteerd moet worden om de functionaliteit van het te bouwen systeem voor klant, ontwerper en ontwikkelaar te beschrijven. Dit heb ik aangeduid met 'functioneel ontwerp'. In de volgende paragrafen geef ik per methode een overzicht van de verschillende onderwerpen die aan de bod komen in die documentatie, hoe die onderwerpen worden gedocumenteerd en wie er bij betrokken zijn.

5.2.1. SDM / LAD: functionele specificatie

SDM / LAD beschrijft functionaliteit in een viertal dossiers. Samen vormen deze dossiers de *functionele specificatie*. Centraal staat het dossier over informatiefuncties. Dit dossier is gericht



Figuur 5.3.: Overzicht van het RUP-proces

op de functies van het informatiesysteem. De andere drie dossiers zijn gericht op gegevens. In tabel 5.1 staat beschreven wat er in de verschillende dossiers wordt vastgelegd.

Bij deze producten onderscheidt SDM / LAD verschillende betrokkenen. Dit zijn de Directe Gebruiker, Indirecte Gebruiker, Automatiseringsdeskundige, Organisatiekundige, Opleidingskundige, Gegevensbeheerder en Verwerkingscentrum. Bij SDM / LAD heeft de automatiseringsdeskundige de rollen ontwerper en ontwikkelaar. De andere betrokkenen horen tot de rol klantmedewerker. Wat opvalt is dus dat SDM / LAD bij het functioneel ontwerp nog geen onderscheid maakt tussen ontwerpers en ontwikkelaars. Verder is er een verfijnd onderscheid in verschillende klantmedewerker-rollen. De verschillende betrokkenen die SDM / LAD onderscheidt bij de functionele specificatie, zijn beschreven in tabel 5.2.

Noemenswaardig is ook het begrip *systeemencyclopedie*. Dit dient tot een "*centrale opslagplaats voor definities van objecten, waaruit kan worden geput bij het vervaardigen van producten. Met behulp van de systeemencyclopedie kunnen deze producten worden gecontroleerd op volledigheid en consistentie*" (Vlasblom et al., 1991, par. 7.2). In het ideale geval zou alle documentatie over het informatiesysteem vanuit deze centrale bron toegankelijk moeten zijn voor alle betrokkenen bij het systeemontwikkelingsproces.

5.2.2. DSDM: functioneel model

DSDM beschrijft functionaliteit in het *functioneel model*. Dit model bevat een prototype dat de gewenste functionaliteit demonstreert en een aantal ondersteunende modellen die ieder een bepaald aspect van het systeem beschrijven. Voor elk afzonderlijk project moet bekeken worden welke modellen daarin nuttig zijn. DSDM vereist dus geen specifieke modellen voor ieder project. Wel is een prototype een vereiste, aangezien op basis daarvan met de klantmedewerkers

Dossier	Beschrijving	Betrokkenen
Informatiefuncties	Beschrijft de door een gebruiker of systeem verrichte functies binnen een informatiesysteem. Belangrijke onderdelen zijn: overwegingen achter het functioneel ontwerp, deelsystemen, informatiefuncties en de functiestructuur (de samenstelling van het informatiesysteem in een hiërarchie van deelsystemen en informatiefuncties).	Directe gebruiker, Indirecte gebruiker, Automatiseringsdeskundige, Organisatiekundige, Opleidingskundige
Functionele Gegevensdefinities	Beschrijft welke gevens het systeem moet vastleggen en hoe die onderling samenhangen, zoals een gebruiker dat ziet. Het bestaat uit: een diagram van het gegevensmodel, een beschrijving van entiteittypen, attributen, de mogelijke waarden voor die attributen, de relaties tussen entiteittypen en het gebruik van entiteittypen door informatiefuncties.	Directe gebruiker, Indirecte gebruiker, Automatiseringsdeskundige, Organisatiekundige, Gegevensbeheerder
Gebruikersinterfaces	Beschrijft de mogelijke interacties van gebruikers met het systeem. Dit zijn beschrijvingen van de menustructuur en de samenstelling en indeling van rapporten van achtergrondfuncties. Voor iedere interactieve informatiefunctie is er een beschrijving van de structuur van de dialoog en de samenstelling en indeling van de schermen van de dialoog.	Directe gebruiker, Automatiseringsdeskundige, Organisatiekundige, Opleidingkundige
Externe interfaces	Beschrijft de interacties van andere systemen met het informatiesysteem. Dit zijn beschrijvingen van de partijen die betrokken zijn bij de interface, afspraken over de ontwikkeling, het gebruik en het beheer van de interface, de structuur van de gegevens die het systeem met andere systemen moet uitwisselen en het protocol hoe de gegevens moeten worden uitgewisseld.	Deelnemers interface, Automatiseringsdeskundige, Organisatiekundige, Verwerkingscentrum

Tabel 5.1.: Onderwerpen functionele specificatie SDM / LAD

Betrokkene	Omschrijving
Directe gebruiker	Directe gebruikers gaan het systeem ook daadwerkelijk gebruiken.
Indirecte gebruiker	Indirecte gebruikers krijgen informatie aangeleverd van gebruikers die het systeem direct gebruiken.
Automatiserings-deskundige	Deze is "*belast met het fysiek ontwerp, de bouw, etc. van het geautomatiseerde deel van het informatiesysteem, dan wel met de gegevensconversie*"
Organisatiekundige	Deze is "*belast met de organisatie-ontwikkeling en het opstellen van de handmatige procedures*
Opleidingskundige	Deze is "*belast met het plannen en realiseren van opleidingen*"
Gegevensbeheerder	Deze doelgroep wordt ook wel data administrator genoemd. Zij zijn "*belast met de functionele coördinatie over de gegevensinfrastructuur*"
Verwerkingscentrum	Dit zijn de deelnemers aan interfaces (voor uitwisseling van data via externe interfaces, (niet-gebruikersinterfaces)).

Tabel 5.2.: Betrokkenen functionele specificatie SDM / LAD

overeenstemming wordt bereikt over de functionaliteit van het te bouwen informatiesysteem. Er zijn vier soorten prototypes te onderscheiden. Overigens is dat onderscheid meestal niet zo strikt als hier opgesomd staat, de verschillende soorten kunnen elkaar ook overlappen (DSDM-Consortium, 2008, http://www.dsdm.nl/version4/2/members/Prototyping.asp). De soorten prototypes zijn:

1. **Business** - laat zien hoe de bedrijfsprocessen geautomatiseerd worden;
2. **Usability** - om te onderzoeken of de gebruikersinterface gebruikersvriendelijk is;
3. **Performance en capability** - om te laten zien dat het systeem goed om kan gaan met intensief gebruik van het systeem;
4. **Capaciteit / techniek** - om te laten zien hoe een bepaalde ontwerpaanpak werkt of om een ontwerpconcept te laten zien.

In de Functioneel Model Iteratie worden voornamelijk het business-, usability en capaciteit / techniekprototypen gebruikt. Deze demonstreren vooral het gedrag / de acties en de interactie met gebruikers. Het bedrijfsdomein en de te verwerken gegevens worden in een apart model beschreven. Daarnaast geeft DSDM een aantal andere modelleringstechnieken aan die gebruikt kunnen worden in de Functioneel Model Iteratie. Er is een onderscheid tussen modellen die samen met het systeem worden opgeleverd (core models) en modellen die alleen tijdens het project gebruikt worden (support models). Voorbeelden van modellen die gebruikt kunnen worden zijn: data flow diagrams (beschrijven de gegevensstroom in een informatiesysteem), use cases (zie tabel 5.4) en system state diagrams (beschrijven de toestanden waarin een systeem kan verkeren). Binnen DSDM worden modellen in samenwerking met klanten opgesteld. Deze modellen zijn zoveel mogelijk gekoppeld aan prototypes.

Bij het functioneel model onderscheidt DSDM verschillende betrokkenen. Dit zijn de Visionary, de Ambassador User, de Advisor User, de Technical co-ordinator en de (Senior) Developer. Ook hierbij valt het op dat DSDM (net als SDM / LAD) bij het functioneel model geen onderscheid maakt tussen ontwerpers en ontwikkelaars. De Senior Developer / Developer heeft de rol van ontwerper en van ontwikkelaar. Daarnaast is er ook hier een verfijnd onderscheid in verschillende klantmedewerker-rollen. De Technical Co-ordinator is niet echt in te delen in één van de drie rollen die bij het functioneel ontwerp betrokken zijn. Zoals de naam aangeeft, is dit meer een projectmanagementrol. De verschillende betrokkenen die DSDM onderscheidt bij het functioneel model, zijn beschreven in tabel 5.3 (DSDM-Consortium, 2008, http://www.dsdm.nl/version4/2/members/Prototyping.asp).

5.2.3. RUP: Software Requirements Specification

RUP beschrijft functionaliteit in de Software Requirements Specification. Dit bestaat uit vier onderdelen: het *use-casemodel*, *use cases*, *supplementaire specificaties* en *storyboards*, zie tabel 5.4. Het use-casemodel dient als communicatiemiddel tussen klantmedewerkers, ontwerpers en ontwikkelaars. Een use case is een beschrijving van de stappen die het systeem uitvoert om een bepaald doel voor een actor te bereiken. Op basis van de requirements wordt het use-casemodel aan het begin van het project opgesteld. Van elke use-case uit het model wordt in detail beschreven hoe het systeem interacteert met de actoren en wat het systeem in die use-case doet. Het use-casemodel is de basis voor alle volgende modellen in het RUP-softwareontwikkelingsproces. In tabel 5.4 zijn de belangrijkste producten van deze fase weergegeven.

Bij het use-casemodel onderscheidt RUP verschillende betrokkenen. Dit zijn de rollen Systeemanalist, Softwarearchitect, Requirements-specificeerder, Stakeholder en Ontwikkelaar. Hier valt op dat er aan de kant van Ontwerpers een verfijnde rolverdeling is. RUP maakt geen verfijnd onderscheid tussen Klantmedewerker-rollen. De verschillende betrokkenen die RUP onderscheidt bij het use-casemodel, zijn beschreven in tabel 5.5.

5.3. Evaluatie van de drie methoden

De besproken methoden beschrijven bijna allemaal dezelfde soorten onderwerpen in het functioneel ontwerp. Wel legt elke methode zijn specifieke nadruk op een bepaald onderwerp, of een bepaalde manier van werken. De onderwerpen in een functioneel ontwerp zijn in een aantal typen onder te verdelen. Dit zijn:

1. Gegevens en domein
2. Gedrag en acties
3. Interactie (met gebruikers en met ander systemen)

Allereerst zal er een beschrijving zijn van de gegevens die moeten worden verwerkt. Het is daarbij nodig om ook het domein te beschrijven, zodat het begrijpelijk is wat die gegevens voorstellen. In het verzekeringen-domein wordt bijvoorbeeld gewerkt met gegevens over polissen. Om beter te begrijpen wat die gegevens voorstellen, is het nuttig om te beschrijven wat een polis in het

Betrokkene	Omschrijving
Ambassador User	De 'ambassador user' komt van het bedrijfsdomein waar het informatiesysteem voor gemaakt wordt. Deze gebruiker is integraal onderdeel van het ontwikkelteam. De 'ambassador user' moet het verlangen, de autoriteit, de verantwoordelijkheid en de kennis hebben om ervoor te zorgen dat het juiste systeem wordt gebouwd voor de organisatie.
Advisory User	De 'advisory user' brengt de praktijkkennis in van de taak die wordt geautomatiseerd. Degene met deze rol zal waarschijnlijk iemand zijn die later het systeem ook gaat gebruiken.
Visionary	De visionair ('visionary') is een gebruikersrol. Deze persoon is degene die normaal gesproken verantwoordelijk is voor het opstarten van het project door enthousiasme en commitment voor het idee en de bedrijfsdoelen. De visionair zou betrokken moeten blijven tijdens het ontwerp en oplevering van het systeem om ervoor te zorgen dat de initiële doelen bereikt worden.
Senior Developer / Developer	Een senior ontwikkelaar ('senior developer') modelleert en interpreteert de gebruikersrequirements en vertaalt die naar prototypes en opleverbare code. Hierbij maakt hij gebruik van het bedrijfsstudie-document en van commentaar van de gebruikers. De senior ontwikkelaar documenteert en ontwikkelt ook de elementen van het systeem die niet in het prototype gevat kunnen worden. Een (junior) ontwikkelaar assisteert met deze taken om zijn (DSDM-)vaardigheden (verder) te ontwikkelen.
Technical co-ordinator	De technische coördinator ('technical co-ordinator') zit boven de afzonderlijke ontwikkelteams. In ieder van die teams is hij maar voor part-time aanwezig. Deze rol zorgt ervoor dat de teams op een consistente manier samenwerken en dat er technische samenhang in het project aanwezig is.

Tabel 5.3.: Betrokkenen functioneel model DSDM

Onderwerp	Beschrijving	Betrokkenen
Use casemodel	Dit model beschrijft de use-cases in samenhang met elkaar en met de actoren van het systeem. Het laat zien wat de functionaliteit van het systeem is en in welke omgeving het systeem staat. Actoren zijn mensen of systemen die met het te ontwikkelen systeem interacteren.	Stakeholders, Systeemanalist
Use cases	Per use case is een beschrijving van de stappen die het systeem uitvoert om een bepaald doel voor een actor te bereiken. Er is onderscheid tussen 'main' (als alles goed gaat), 'alternate' (als er een fout optreedt, maar hersteld wordt) en 'exception' (als het fout gaat en niet hersteld wordt) uitvoering van die stappen. In zijn simpelste vorm kan een use case aangeven wat voor interactie een actor met het systeem kan hebben.	Stakeholders, Systeemanalist
Supplementaire specificaties	Dit zijn beschrijvingen van additionele vereisten aan het informatiesysteem die niet in use-cases uit te drukken zijn, maar wel nodig om het systeem te kunnen bouwen naar wens van de klant.	Software-architect, Requirements specificeerder, Systeemanalist
Storyboards	Dit zijn schematische diagrammen van de schermen uit de applicatie. De storyboards zijn gekoppeld aan use-cases.	Stakeholders, Systeemanalist

Tabel 5.4.: Onderwerpen in een Software Requirements Specification

Betrokkene	Omschrijving
Systeemanalist	Deze stelt in samenwerking met de stakeholders de requirements en het user interface-ontwerp op en heeft kennis van het bedrijfsdomein.
Softwarearchitect	Deze bewaakt de integriteit van de verschillende use-cases en is verantwoordelijk voor de belangrijke technische beslissingen.
Requirements-specificeerder	Deze zet de requirements om in een use-case-ontwerp en werkt de use cases uit.
Stakeholder	Een stakeholder heeft een belang bij de uitkomst van het project. De stakeholder geeft input aan het begin van het project door de requirements op te stellen en geeft feedback op de opgeleverde modellen en prototypes.
Ontwikkelaar	Op basis van de de Software Requirements Specification maakt de ontwikkelaar een aantal modellen om het (technische) ontwerp van het informatiesysteem te beschrijven. Vervolgens maakt de ontwikkelaar het informatiesysteem op basis van deze modellen.

Tabel 5.5.: Betrokkenen use-casemodel RUP

verzekerings-domein voorstelt. Het tweede punt laat zien wat er met de gegevens gedaan kan worden; welke functies het informatiesysteem op de gegevens kan uitvoeren. Als laatste is het van belang te beschrijven hoe gebruikers en andere systemen met het informatiesysteem kunnen interacteren. Een volledig geïsoleerd systeem heeft in het algemeen niet veel waarde.

Wat erg belangrijk is, is dat het informatiesysteem aansluit bij de processen in het bedrijf. Alle drie de methoden beschrijven de bedrijfsprocessen vóórdat het functioneel ontwerp wordt gemaakt (SDM / LAD in de definitiestudie, DSDM in de bedrijfsstudie, RUP bij de bedrijfsmodellering). Het functioneel ontwerp kan dan eventueel verwijzen naar die beschrijving. De beschrijving van bedrijfsprocessen wordt in het functioneel ontwerp ook gebruikt om te verantwoorden waarom welke functies worden gemaakt.

Zoals gezegd heeft ieder van de methoden zijn specifieke focus om functionaliteit te communiceren tussen klant, ontwerper en ontwikkelaar.

SDM/LAD legt de nadruk vooral op een *document-gerichte* communicatiewijze. SDM/LAD onderscheidt functie-gerichte documenten en gegevens-gerichte documenten. Deze documentatiewijze is vergelijkbaar met de wijze waarop Quinity haar functioneel ontwerp documenteert. Door deze documenten wordt er gecommuniceerd tussen de drie betrokken rollen. De documenten zouden in een systeemencyclopedie moeten worden opgeslagen zodat de consistentie en volledigheid van de documenten verzekerd is.

DSDM hanteert een *prototype-gerichte* communicatiewijze. Door de prototypes stemt een ontwerper (in samenwerking met een ontwikkelaar) met de klantmedewerker af wat het systeem moet kunnen en hoe de interactie met het systeem er uit komt te zien. In het ideale geval is er een prototype dat steeds verder uitgebreid wordt totdat uit het prototype het informatiesysteem ontstaat waarmee de klant kan gaan werken. Naast de prototypes kunnen er ook tal van

modellen worden gebruikt die ieder een bepaald aspect van het systeem belichten. Hieronder vallen domein-, gedrags-, gegevens- en systeeminteractiemodellen.

RUP heeft een *model-gerichte* communicatiewijze. De modellen belichten ieder verschillende aspecten van het systeem. Het use casemodel is hierbij de basis van alle andere modellen, het beschrijft het gedrag van het systeem. Verder zijn er net als bij DSDM domein-, gedrags-, gegevens- en systeeminteractiemodellen. Door de model-gerichte communicatiewijze kan er goed overzicht gehouden worden over de ontwerpdocumentatie. De verschillende UML-modellen kunnen aan elkaar gekoppeld worden en in verschillende 'views' worden bekeken.

Elke methode maakt een verfijnd onderscheid in de rollen die betrokken zijn bij het functioneel ontwerp. SDM / LAD en DSDM maken daarbij een verfijnd onderscheid in klantmedewerker-rollen. Blijkbaar is het nuttig dat er verschillende soorten klantmedewerkers betrokken zijn bij het functioneel ontwerp. Iedere soort klantmedewerker levert kennis vanuit een bepaalde invalshoek. Dit zorgt ervoor dat er een informatiesysteem ontworpen wordt dat gebaseerd is op de eisen en wensen van een brede groep.

Naast dat een functioneel ontwerp gemaakt wordt op basis van een diverse klantmedewerker-groep, is het belangrijk dat een functioneel ontwerp het gedrag van het gewenste informatie-systeem goed beschrijft. Dit gedrag moet zodanig beschreven zijn, dat de verschillende soorten klantmedewerkers kunnen controleren of het functioneel ontwerp beschrijft wat zij bedoelen. Daarnaast moeten ook ontwikkelaars begrijpen hoe zij het ontwerp moeten vertalen naar het informatiesysteem. In alle drie de methoden is er aandacht voor de documentatie hiervan. SDM / LAD beschrijft dit in de dossiers informatiefuncties en gebruikersinterfaces, DSDM in het pro-totype (mogelijk in samenwerking met een aantal modellen) en RUP in het use-casediagram met de verschillende uitgewerkte use-cases. De manier waaróp het functioneel ontwerp wordt gedocumenteerd, is voor ieder van de drie methoden wel verschillend. Het is moeilijk aan te geven welke manier nu het beste is voor de communicatie. Wat wel een voordeel is bij DSDM en RUP, is dat het wordt aangeraden om modellen te maken die onderling met elkaar verbonden zijn. Hierdoor is het eenvoudiger om te zien hoe een bepaald vaag idee leidt tot een concrete implementatie van dat idee. De eisen en wensen van de klant (de requirements) zijn daardoor dus beter traceerbaar, en de documentatie van het systeem is beter doorzoekbaar.

6. Voorstel documentatiewijze

Om op een goede manier om te kunnen gaan met grote hoeveelheden documentatie, wordt in dit hoofdstuk een voorstel gedaan voor een documentatiewijze van de functioneel ontwerpdocumentatie bij Quinity. Dat gebeurt aan de hand van de eisen die zijn geformuleerd in paragraaf 4.2. De ideeën die zijn opgedaan bij de vergelijking van drie systeemontwikkelingsmethoden in hoofdstuk 5 zijn in het documentatievoorstel verwerkt. Dit betreft met name de modelgebaseerde aanpak van documentatie. Hierdoor is het werken met de ontwerpdocumentatie goed te ondersteunen met tools.

Eerst wordt in paragraaf 6.1 voor ieder van de vier eisen aangegeven wat daarvoor aan de functioneel ontwerpdocumentatie moet worden aangepast of toegevoegd. Het voorstel komt neer op structurering van de ontwerpdocumentatie volgens het functioneel ontwerpmodel. Dit model beschrijft wat voor informatie er in de documentatie kan worden vastgelegd, en hoe die informatie met elkaar samenhangt. De aanpassingen en toevoegingen op het model zijn gebaseerd op inzichten uit de literatuur. Paragraaf 6.2 gaat in op een aantal risico's van de voorgestelde documentatiewijze. De laatste paragraaf, paragraaf 6.3, beschrijft een concrete gebruiksmogelijkheid die op basis van dit documentatievoorstel te bereiken is.

6.1. Aanpassingen op het functioneel ontwerpmodel

Deze paragraaf beschrijft de aanpassingen aan het functioneel ontwerpmodel die nodig zijn om de eisen uit paragraaf 4.2 te kunnen bereiken. Per eis uit het praktijkonderzoek is aangegeven wat daarvoor nodig is.

6.1.1. Onderhoudbaarheid

De eerste eis is een goede *onderhoudbaarheid* van de functioneel ontwerpdocumentatie. Een goede onderhoudbaarheid van de documentatie is een belangrijke factor voor de kwaliteit van een informatiesysteem als geheel (Broy et al., 2006). Van doorslaggevend belang is daarbij dat de documentatie actueel, samenhangend en zo min mogelijk redundant is. Tegelijkertijd blijkt uit de praktijk dat dit erg moeilijk te bereiken is. Documentatie is lang niet altijd actueel en consistent, maar hoewel dat zo is, kan de documentatie toch wel relevant zijn voor een informatiesysteem (Forward en Lethbridge, 2002).

Om de functioneel ontwerpdocumentatie consistent te houden, moet er een goede ondersteuning van verwijzingen zijn. Dat is nodig om de samenhang tussen de verschillende onderdelen van de documentatie aan te kunnen geven. De verwijzingen moeten de relevante onderdelen in de functioneel ontwerpdocumentatie *uniek kunnen identificeren*. Hiervoor moet bekend zijn wat

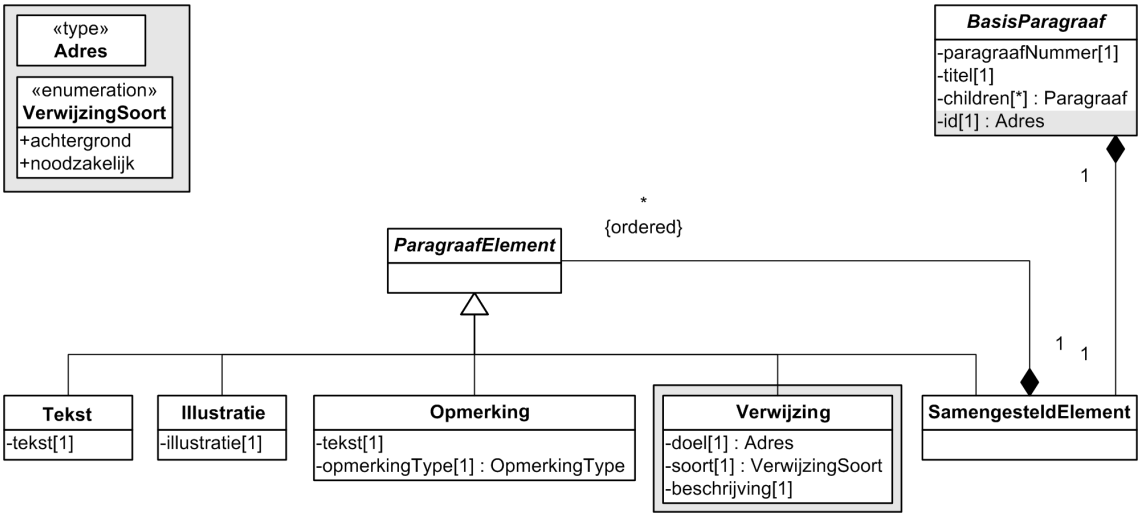
een 'relevant onderdeel' in de documentatie is. In het functioneel ontwerpmodel uit hoofdstuk 3 zijn de verschillende onderdelen van een functioneel ontwerp beschreven. Daaronder vallen in ieder geval alle documentproducten binnen een functioneel ontwerp. Daarnaast zijn er nog de verschillende paragrafen binnen functionele beschrijvingen en de entiteiten en attributen uit het datamodel waarnaar een verwijzing kan verwijzen. De relevante onderdelen om naar te verwijzen in de ontwerpdocumentatie zijn dus de producten van functioneel ontwerp, de paragrafen in functionele beschrijvingen en de attributen en entiteiten uit het datamodel (zie het model in figuren 3.9, tot en met 3.12).

Voor de unieke identificatie van deze onderdelen is een adresseringsschema nodig waarmee die onderdelen uit de functioneel ontwerpdocumentatie geadresseerd kunnen worden. Een Verwijzing (zie figuur 3.6 uit het functioneel ontwerpmodel) kan dan gaan naar een adres dat is gespecificeerd volgens dat adresseringsschema. Dit is weergegeven in figuur 6.1. De informatie die is toegevoegd of aangepast is weergegeven met een grijze achtergrond. Het type Adres specificeert de mogelijke adressen volgens het adresseringsschema. De BasisParagraaf is uitgebreid met het attribuut 'id' van het type Adres. Zodoende kan een Verwijzing dan verwijzen naar een paragraaf. Daarnaast krijgen de producten van functioneel ontwerp en de attributen en entiteiten uit het datamodel ook een attribuut 'id' van het type Adres (zie figuren 6.2 en 6.3).

Naast het Adres-type, is in figuur 6.1 ook het type 'VerwijzingSoort' opgenomen. Dit type specificeert soort van een Verwijzing. Er zijn twee mogelijke waarden voor VerwijzingSoort: 'achtergrond' en 'noodzakelijk'. De waarde 'achtergrond' geeft aan dat de verwijzing achtergrondinformatie geeft over de informatie waarvandaan de verwijzing gedaan wordt. De waarde 'noodzakelijk' geeft aan dat de verwijzing verwijst naar noodzakelijke informatie om de rest van de paragraaf waar de verwijzing in staat te begrijpen. Dit onderscheid tussen achtergrondinformatie en noodzakelijke informatie kan gebruikt worden om op verschillende manieren met de twee typen verwijzingen om te kunnen gaan. Een Verwijzing kan verder een tekstuele beschrijving bevatten van de reden waarom de Verwijzing is opgenomen. Deze tekstuele beschrijving wordt opgenomen in het attribuut 'beschrijving'.

Verwijzingen moeten *robuust* zijn. Gedurende het bestaan van een functioneel ontwerp worden er regelmatig dingen aan toegevoegd, verwijderd of aangepast. Verwijzingen kunnen stuk gaan doordat ze verwijzen naar een onderdeel dat is verwijderd, of doordat een onderdeel is verplaatst. Ook kan het zijn dat de verwijzing inhoudelijk niet meer klopt; het document of de paragraaf bestaat nog wel, maar de inhoud daarvan is zodanig aangepast dat de verwijzing daarnaar niet meer de oorspronkelijke bedoeling weergeeft. In de database-wereld staat dit bekend als 'referentiële integriteit': het databasesysteem zorgt ervoor dat de consistentie van de database gewaarborgd blijft. Robuuste verwijzingen zijn dus verwijzingen die zorgen dat de referentiële integriteit van de documentatie gewaarborgd blijft.

Om te zorgen dat verwijzingen robuust zijn, is het het beste om de documentatie *via één plaats toegankelijk* te maken. In zo'n repository, of systeemencyclopedie, kan dan bijgehouden worden welke verwijzingen er allemaal in de ontwerpdocumentatie voorkomen. Wanneer de objecten (paragrafen of producten) waarnaar verwezen word aangepast over verwijderd worden, kan het systeem dat detecteren en zo nodig de ontwerper waarschuwen voor die wijzigingen. Als een ontwerper een object verwijdert, kan het systeem aangeven dat dat niet mag, omdat er nog een verwijzing naar het te verwijderen object bestaat. Of bij een aanpassing van een object kan



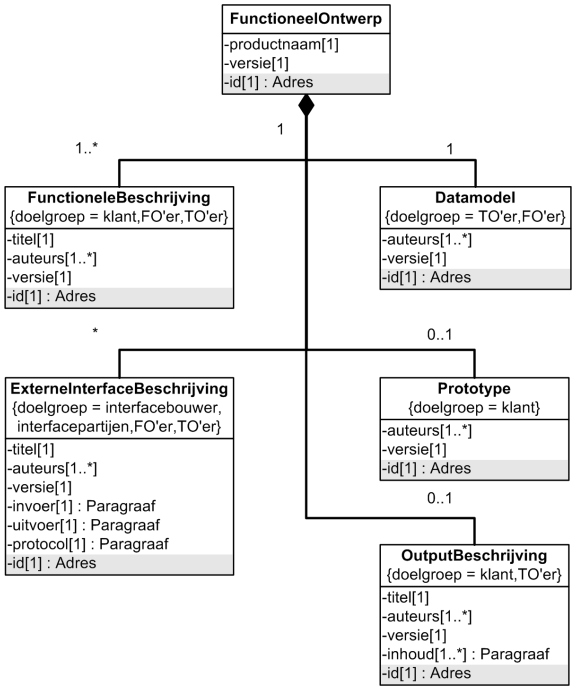
Figuur 6.1.: De BasisParagraaf uitgebreid met adressering.

een systeem aangeven dat de paragrafen die verwijzen naar dat object mogelijk ook moeten worden aangepast. In de repository kunnen alle controles op consistentie en verwijzingen tussen de onderdelen van functioneel ontwerpen plaatsvinden.

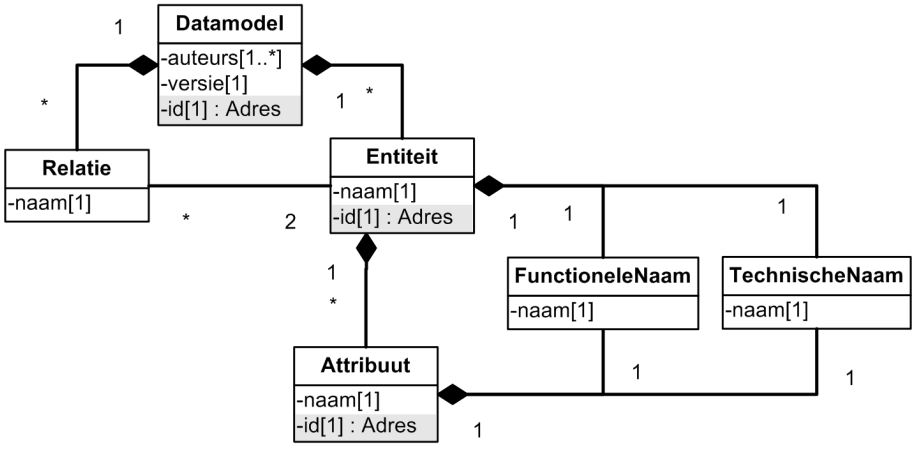
6.1.2. Traceerbaarheid

De tweede eis is dat de requirements traceerbaar zijn in functioneel ontwerp, technisch ontwerp en code: de *traceerbaarheid*. In de literatuur wordt het belang van traceerbaarheid van requirements naar de verschillende onderdelen van de softwaredocumentatie ook onderstreept (Ramesh, 1998; Spanoudakis, 2002; Dick, 2005). Bij de traceerbaarheid van requirements naar de softwaredocumentatie zijn verwijzingen van belang. Eén of meer onderdelen in het functioneel ontwerp zijn verbonden met één of meer requirements. Evenzo zijn één of meer onderdelen in het technisch ontwerp verbonden met één of meer onderdelen in het functioneel ontwerp. En één of meer onderdelen van de code zijn verbonden met één of meer onderdelen in het technisch ontwerp. Zo zijn er over de hele keten tussen requirements en code verbindingen die aangeven wat op welke plaats is gespecificeerd, toegelicht of gerealiseerd. Deze 'traceerbaarheid-verwijzingen' geven in feite de verantwoording aan van de reden waarom bepaalde onderdelen in een functioneel ontwerp, technisch ontwerp of code zijn opgenomen. Het is dus belangrijk dat de verbindingen in die keten tussen requirements en code, ofwel de traceerbaarheid, op een goede manier vastgelegd en onderhouden kunnen worden.

Om verbindingen tussen de verschillende onderdelen van softwaredocumentatie aan te kunnen geven, is het nodig dat de onderdelen van functioneel ontwerpdocumentatie *uniek te identificeren* zijn. Daarnaast moeten ook de requirements, de verschillende onderdelen in de technisch ontwerpdocumentatie en onderdelen in de code uniek te identificeren zijn. Hiervoor is het nodig (net als bij de functioneel ontwerpdocumentatie) dat de relevante onderdelen in die producten bekend zijn. Een model van die producten is dus ook gewenst. Verwijzingen tussen deze producten van systeemontwikkeling zijn met hetzelfde verwijzingsmechanisme te realise-



Figuur 6.2.: Adressering in de producten van functioneel ontwerp



Figuur 6.3.: Adressering in het datamodel

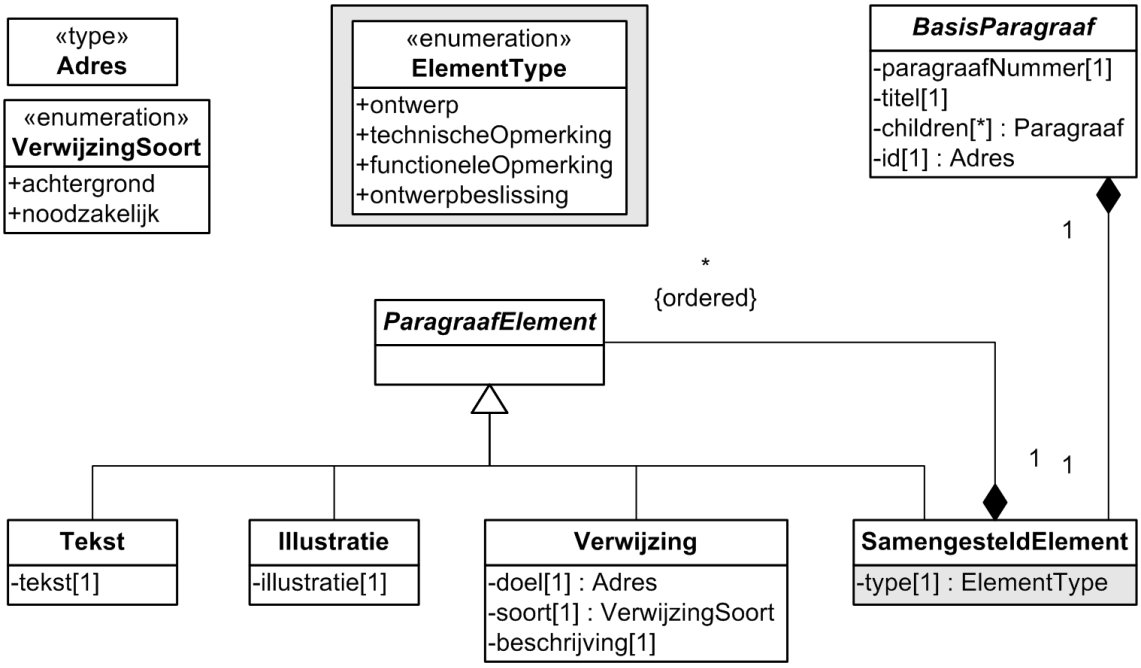
ren als verwijzingen binnen de functioneel ontwerpdocumentatie (uit paragraaf 6.1.1). Naast de informatie die bij een gewone Verwijzing opgenomen wordt (de attributen 'doel', 'soort' en 'beschrijving') kunnen er ook andere eigenschappen over de traceerbaarheidsverwijzing worden gedocumenteerd zoals status, gebruikshistorie en aanmaakdata (Marcus et al., 2005). De 'traceerbaarheids-verwijzingen' zijn uitsluitend tussen (onderdelen van) producten uit verschillende fasen van systeemontwikkeling (bijvoorbeeld tussen een requirementsdocument en een functioneel ontwerp of tussen een functioneel ontwerp en een technisch ontwerp). Er zijn nooit 'traceerbaarheids-verwijzingen' tussen onderdelen van producten uit dezelfde fase.

Omdat er geen model is van requirements en technisch ontwerpdocumentatie, is het moeilijk om de mogelijke verwijzingen tussen functioneel ontwerp en die twee producten aan te geven. Toch kan er wel iets over gezegd worden. Door ook in requirements en technisch ontwerpdocumentatie hetzelfde adresseringsschema te gebruiken als in de vorige paragraaf is voorgesteld, kunnen verwijzingen tussen deze producten worden opgenomen. Het gereedschap dat deze verwijzingen aanbrengt zal rekening moeten houden met de toegestane verwijzingen tussen requirements, functioneel ontwerp, technisch ontwerp en code. Functionele requirements zullen gekoppeld zijn aan informatiefuncties in het functioneel ontwerp. Alle functionele requirements zullen uiteindelijk aan een onderdeel in het functioneel ontwerp gekoppeld moeten zijn. Non-functionele requirements zijn helemaal niet aan het functioneel ontwerp gekoppeld, maar zijn direct gekoppeld aan het technisch ontwerp. Uitsluitend als een non-functionele requirement van invloed is op een bepaalde functie, dan kan er ook een koppeling naar het functioneel ontwerp zijn.

6.1.3. Annotaties

De derde eis is het opnemen van annotaties in de functioneel ontwerpdocumentatie. In figuur 6.4 is de noodzakelijke aanpassing van het functioneel ontwerpmodel hiervoor weergegeven.

Metadata is nodig om computersystemen te laten 'begrijpen' met wat voor informatie ze omgaan. Een annotatie is ook een vorm van metadata. De hier beschreven annotaties geven een computersysteem informatie over het type annotatie. Voor een mens is een annotatie een toelichting op het ontwerp (Hodge, 2004). Om annotaties op te kunnen nemen moet er extra informatie in het functioneel ontwerpmodel worden opgenomen waarmee die annotaties beschreven worden. Er moet ook onderscheid gemaakt kunnen worden tussen verschillende categorieën annotaties. Daarmee kan een betrokkene bij het functioneel ontwerp er voor kiezen om bepaalde categorieën wel of niet weer te geven op het medium (bijvoorbeeld scherm of print) waarop de betrokkene de functioneel ontwerpdocumentatie bekijkt. Functioneel ontwerpdocumentatie moet dus een opslagstructuur hebben waarin per relevant onderdeel aangegeven kan worden over welk soort informatie het gaat: wel of geen annotatie en de verschillende categoriën annotaties. Een relevant onderdeel is hier een ParagraafElement, zoals in het functioneel ontwerpmodel beschreven (zie figuur 3.6). In het model zou het paragraafelement 'Opmerking' vervangen kunnen worden door een attribuut 'type' bij de klasse 'SamengesteldElement'. Dit attribuut kan dan de waarden 'ontwerp', 'functioneleOpmerking', 'technischeOpmerking' en 'ontwerpbeslissing' hebben. Wanneer er meer annotatietypen nodig zijn, kan deze lijst uitgebreid worden. Standaard heeft een SamengesteldElement het type 'ontwerp', waarmee is aangegeven dat ParagraafElementen direct onder dit SamengesteldElement normale ontwerp-



Figuur 6.4.: Type-informatie voor een SamengesteldElement

informatie is. De andere typen geven annotaties aan. Dit zijn respectievelijk een functionele opmerking, een technische opmerking en een ontwerpbeslissing. Wanneer een betrokkene aangeeft geen technische opmerkingen te willen zien in de ontwerpdocumentatie, dan verbergt het systeem alle SamengesteldeElementen en onderhangende ParagraafElementen met het type 'technischeOpmerking'.

6.1.4. Intelligente zoekmogelijkheden

De vierde eis is het hebben van intelligente zoekmogelijkheden in de functioneel ontwerpdocumentatie. Door de documentatie volgens het functioneel ontwerpmodel op te slaan, wordt de documentatie relationeel gestructureerd. Aan de andere kant bevat de documentatie ook ongestructureerde informatie in de vorm van tekst en illustraties. De functioneel ontwerpdocumentatie is dus een vorm van *semi-structured data*(Buneman, 1997). In dit soort data kan worden gezocht met een kruising van information retrieval methoden en gestructureerde querytalen.

In de documentatie moet informatie worden opgenomen over typering en versionering van de informatie die in een ontwerp kan worden vastgelegd. Daarmee kan dan worden gezocht op een bepaald type informatie (zoals 'begrippen' of 'informatiefuncties') of een bepaalde versie van informatie binnen de documentatie. Het functioneel ontwerpmodel zelf zorgt voor typering van de verschillende paragrafen. Beschrijvingen in een InformatiefunctieParagraaf gaan over informatiefuncties, beschrijvingen in een BegripPararaaf gaan over begrippen. Hiermee zit de typering van de informatie in het functioneel ontwerpmodel ingebakken. Informatie die volgens de structuur van het functioneel ontwerpmodel is vastgelegd, is dus automatisch getypeerd.

Door die typering kan een zoeksysteem 'intelligent' informatie extraheren voor het maken van een zoekindex. Tekst die is vastgelegd in een InformatiefunctieParagraaf, gaat blijkbaar over een informatiefunctie, tekst in een BegripParagraaf gaat blijkbaar over een begrip. Of wanneer er veel verwijzingen naar een bepaalde paragraaf zijn, is die paragraaf blijkbaar belangrijk.

Naast de typering is ook versie-informatie aangebracht in de producten van functioneel ontwerp. Een zoeksysteem kan op basis hiervan mogelijke gevonden relevante antwoorden op een zoekvraag filteren. Het systeem zal alleen die resultaten teruggeven die een versienummer hebben die binnen de beperkingen liggen die de gebruiker heeft opgegeven.

Intelligente zoekmogelijkheden kunnen dus bereikt worden een zoeksysteem te maken dat gebruik maakt van de structuur en metadata die het in het functioneel ontwerpmodel is gedefinieerd.

6.2. Risico's

Er is een aantal risico's wanneer de functioneel ontwerpdocumentatie wordt opgeslagen volgens de eisen uit de vorige paragraaf. Ten eerste bestaat de vrees dat functioneel ontwerpers hun creativiteit niet meer in het functioneel ontwerp kwijt kunnen. Ontwerpers willen graag zo vrij mogelijk zijn in wat ze opschrijven en hoe ze dat doen. Als de structuur van de functioneel ontwerpdocumentatie echter goed is, zal dit geen probleem zijn. De structuur is juist nodig zodat er beter met elkaar samengewerkt kan worden. Door de structuur wordt de documentatie namelijk beter onderhoudbaar en beter doorzoekbaar.

De kracht van het functioneel ontwerpmodel komt naar voren wanneer de juiste gereedschappen worden ingezet die met dat model kunnen omgaan. Een tweede risico is echter dat er teveel vertrouwen gesteld wordt in het gebruik van tools. Uitgangspunt moet echter zijn dat tools slechts hulpmiddelen zijn die een ontwerper kan gebruiken bij zijn werk. Een tool lost niet alle problemen op, maar kan wel ondersteuning bieden op vlakken waar een computer goed in is (omgaan met veel gestructureerde data).

Een derde risico is dat het te ingewikkeld is om met de tool te werken. Ontwerpers zijn dan meer tijd dan nu kwijt om dingen vast te leggen. Het moet dus eenvoudig zijn om een tool te gebruiken. Liefst is het ook eenvoudig om aan te leren hoe je met een tool kunt werken. Dat komt de inzet van zo'n tool in het ontwerpproces ten goede.

Een belangrijk punt is verder dat functionele ontwerpen geversioneerd kunnen worden. Het versiebeheer is nodig om terug te kunnen kijken naar een moment eerder in het proces waarin een functioneel ontwerp is opgesteld. Wijzigingen kunnen zo eventueel ongedaan worden gemaakt. De vraag is wel op welke onderdelen uit het functioneel ontwerp het versiebeheer moet worden toegepast. Op dit moment is dat alleen op de producten van functioneel ontwerp. Misschien kan het ook nuttig zijn om versiebeheer te verfijnen tot op paragraafniveau. In ieder geval kunnen bij een onderdeel op hoger niveau meerdere versies van producten op lager niveau horen. In figuur 3.14 staat bijvoorbeeld het functioneel ontwerp 'Polisadministratie klant X', met versienummer 0.9. Dit functioneel ontwerp bevat de functionele beschrijvingen 'BeherenPolissen', met versienummer 0.2, en 'WerkenMetVolmachten', met versienummer 0.1.5. Een functioneel ontwerp kan dus producten onder zich hebben die zich ieder in een andere versie

bevinden. Omdat vanuit deze producten naar elkaar verwezen wordt, is het wel belangrijk dat in de adressering van een verwijzing rekening gehouden wordt met het versiebeheer.

6.3. Gebruiksmogelijkheden

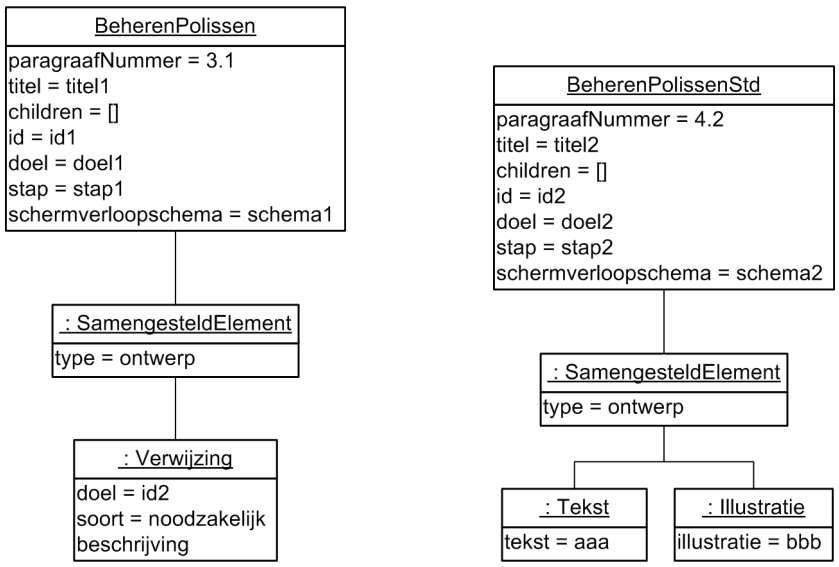
Het functioneel ontwerpmodel uit hoofdstuk 3 beschrijft de structuur van de functioneel ontwerpdocumentatie zoals die nu is. Samen met de voorgestelde aanpassingen uit dit hoofdstuk, zijn er een aantal concrete gebruiksmogelijkheden. Het functioneel ontwerpmodel is te gebruiken als input bij het maken of uitkiezen van tools voor functioneel ontwerp dat deze gebruiksmogelijkheden kunnen realiseren. In deze paragraaf werk ik twee van die mogelijkheden uit. Daarnaast beschrijf ik kort een aantal andere gebruiksmogelijkheden.

6.3.1. Embedding links

Op basis van de VerwijzingSoort kan een tool op verschillende manieren met een Verwijzing om gaan. Normaal gesproken zal er op de plaats van de Verwijzing een stukje tekst opgenomen worden dat aangeeft waar de Verwijzing naar toe wijst. Dit kan bijvoorbeeld het gedrag zijn wanneer de VerwijzingSoort 'achtergrond' is. Als de VerwijzingSoort 'noodzakelijk' is, zou een tool er voor kunnen kiezen om hetgene waarnaar verwezen wordt direct op de plaats van de Verwijzing weer te geven. Dit is een 'embedding link'. We spreken af dat een Verwijzing met VerwijzingSoort 'noodzakelijk' alleen mag verwijzen naar Paragrafen. Als dat niet het geval zou zijn, zou een heel document opgenomen moeten worden op de plaats van de Verwijzing en dat wordt problematisch.

In de praktijk kan dit het volgende inhouden. Veel functioneel ontwerpdocumentatie gaat over een systeem dat gebruik maakt van standaardcomponenten. De standaardcomponenten (bijvoorbeeld componenten van QIS) zijn beschreven in verschillende functionele beschrijvingen, datamodellen, externe interfacebeschrijvingen en prototypes. Wanneer een systeem gebruik maakt van standaardcomponenten, wordt alleen de maatwerk-functionaliteit beschreven. Dit gebeurt in een losstaande functionele beschrijving die hoort bij dat systeem. Als er geen uitzondering is op de standaardcomponent-functionaliteit, kan een functioneel ontwerper in de functionele beschrijving van het maatwerk de beschrijving voor een bepaalde functionaliteit rechtstreeks uit de functionele beschrijving van de standaardcomponent halen. Dit geeft de ontwerper dan aan met een Verwijzing waarvan de VerwijzingSoort 'noodzakelijk' is. Een tool kan deze Verwijzingen naar wens weergeven als embedding links (dus geen kopie), de beschrijving blijft maar op één plaats vastgelegd. Voor een klant is dat handig zodat hij niet telkens in de functionele beschrijving van de standaardcomponent hoeft te bladeren. Voor een technisch ontwerper kan de tool die 'embedding' weglaten, omdat de ontwerper alleen de uitzondering op standaardfunctionaliteit hoeft te zien.

Ik zal de embedding links toelichten met een voorbeeld. In figuur 6.5(a) is de InformatiefunctieParagraaf 'Beheren polissen' uit een maatwerk-functionele beschrijving weergegeven. Daarnaast staat in figuur 6.5(b) een InformatiefunctieParagraaf uit een functionele beschrijving van een standaardcomponent: BeherenPolissenStd. Zoals te zien in figuur 6.5(a), is een Verwijzing opgenomen. Het doel van de Verwijzing is 'id2', de InformatiefunctieParagraaf uit figuur 6.5(b),



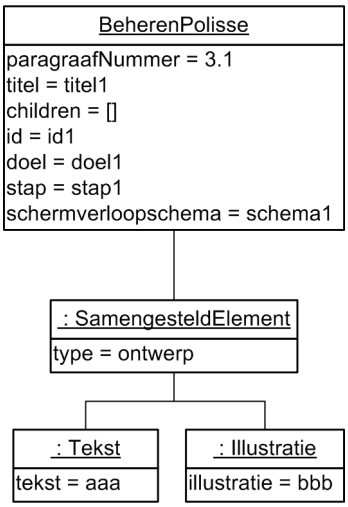
(a) Fragment uit maatwerk

(b) Fragment uit standaardcomponent

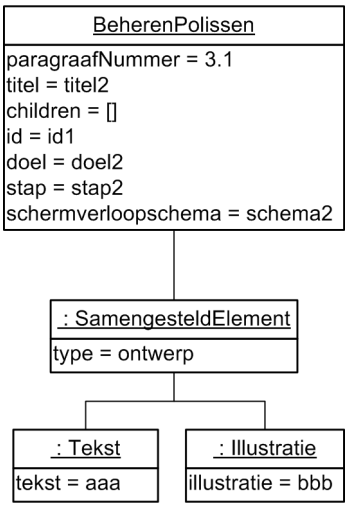
Figuur 6.5.: Fragmenten uit twee functionele beschrijvingen

en het type van deze verwijzing is 'noodzakelijk'. Wanneer de functionele beschrijving van het maatwerk wordt weergegeven door een tool, kunnen Verwijzingen met het type 'noodzakelijk' dus als embedded link worden opgenomen. Dit resulteert in dit geval in de structuur die is weergegeven in figuur 6.6. De Tekst en Illustratie van 'BeherenPolissenStd' is dus in plaats van de Verwijzing komen te staan. Voor de duidelijkheid: deze structuur is alleen een tijdelijke weergave. Er wordt niets aan de functioneel ontwerpdocumentatie aangepast. De documentatie wordt alleen real-time op een andere manier weergegeven.

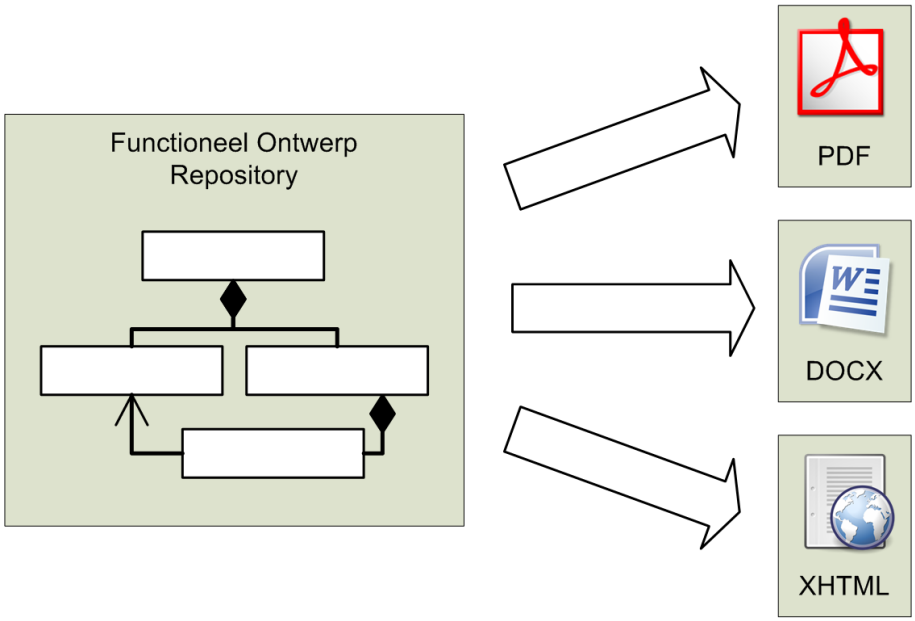
Nu is er echter nog wel een probleem dat is blijven liggen. Want in figuur 6.6 zijn niet de titel, children, doel, stap en schermverloopschema overgenomen uit BeherenPolissenStd. Wanneer er verder geen aanpassingen zijn op de standaardfunctionaliteit, van BeherenPolissenStd, moeten deze paragrafen zonder meer overgenomen worden. Dit kan opgelost worden door in de paragrafen van 'children', 'doel1', 'stap1' en 'schema1' weer Verwijzingen (met Verwijzing-Soort 'noodzakelijk') op te nemen naar de betreffende Paragrafen uit 'BeherenPolissenStd' (de paragrafen van 'children', 'doel2', 'stap2' en 'schema2'). Maar dat is een erg arbeidsintensieve oplossing en dat willen we juist niet. De tool zal dus een optie moeten bieden om onderliggende paragrafen mee te nemen in de embedding. In het geval van InformatiefunctieParagrafen zijn de onderliggende paragrafen de children-paragrafen, de doel-paragraaf, de stap-paragraaf en de schermverloopschema-paragraaf. Een andere optie kan aangeven of de titel van een paragraaf moet worden overgenomen. De eerste vorm van embedding noem ik eenvoudige embedding. De tweede vorm noem ik uitgebreide embedding. In figuur 6.7 is de uitgebreide embedding weergegeven van de Verwijzing uit figuur 6.5(a).



Figuur 6.6.: Structuur van BeherenPolissen na eenvoudige embedding



Figuur 6.7.: Structuur van BeherenPolissen na uitgebreide embedding



Figuur 6.8.: Mogelijke transformaties van functioneel ontwerpdocumentatie

6.3.2. Genereren van documentatie

Door de beschreven structurering van de functioneel ontwerpdocumentatie zijn er nog meer mogelijkheden met de vernieuwde functioneel ontwerpdocumentatie. Een mogelijkheid is de transformatie van functioneel ontwerpdocumentatie in verschillende vormen. De structuur van de documentatie is zodanig flexibel dat het eenvoudig is om te transformeren naar andere structuren. Dat kan van pas komen wanneer de documentatie in verschillende vormen moet worden weergegeven. Soms wil je de documentatie als een document weergeven, een andere keer kan het misschien makkelijker zijn om de documentatie als een webpagina weer te geven. Daarbij kan het gaan om verschillende bestandsformaten zoals PDF, DOC(X) en (X)HTML. Vanuit een functioneel ontwerprepository zouden deze structuren gegenereerd kunnen worden wanneer de juiste transformatieregels daarvoor gedefinieerd zijn (zie figuur 6.8). Door de structurering kunnen functionele ontwerpen ook een uniforme lay-out krijgen. Een ontwerper beschrijft het ontwerp en specificeert *wat* er wordt vastgelegd (informatiefuncties, begrippen, verwijzingen, etc.). *Hoe* de beschrijving van het ontwerp er uit moet zien regelt het systeem op basis van stijlregels voor de verschillende onderdelen die een functioneel ontwerp kan hebben.

6.3.3. Andere gebruiksmogelijkheden

Naast de embedding links en het genereren van documentatie zijn er andere gebruiksmogelijkheden. Te denken valt aan:

- Traceability checker: controleert of onderdelen uit een functioneel ontwerp nog verwijzen naar iets uit het bijbehorende technisch ontwerp en of onderdelen uit een technisch ontwerp nog verwijzen naar iets uit het bijbehorende functioneel ontwerp.

- Afhankelijkheden checker: controleert welke afhankelijkheden en koppelingen er zijn in ontwerpdocumentatie. Veel afhankelijkheden en koppelingen tussen twee onderdelen zou een indicator kunnen zijn van een verkeerde indeling van het functioneel ontwerp. Die verschillende onderdelen zijn blijkbaar moeilijk los van elkaar uit te leggen. Het functioneel ontwerp wordt duidelijker als beide onderdelen samen worden uitgelegd en niet in aparte onderdelen.
- Intelligent wijzigen: als een ontwerper een informatiefunctie wijzigt, zou de ontwerper, voor alle onderdelen die naar die informatiefunctie wijzen, na willen gaan of die onderdelen ook gewijzigd moeten worden. Het systeem zou kunnen aangeven welke onderdelen allemaal verwijzen naar de informatiefunctie die je hebt gewijzigd.
- Intelligent zoeken: doordat het systeem weet welk type de verschillende onderdelen in een functioneel ontwerp hebben (het weet wat een informatiefunctie representeert, wat een begrip is, wat een functionele beschrijving is, wat een externe interfacebeschrijving is, etc.) kan het systeem intelligenter zoeken door alleen in een bepaalde subset van de documentatie te zoeken. Ook zou bij het zoeken rekening gehouden kunnen worden met versie-informatie: bijvoorbeeld alleen in een bepaalde versie van een functioneel ontwerp zoeken, of alleen in de laatste versie van meerdere functionele ontwerpen.
- Zichtbaarheid van metadata: ontwerpbeslissingen en functionele of technische opmerkingen kunnen door het systeem met een druk op de knop naar wens al dan niet worden weergegeven.
- Traceability van requirements: het systeem houdt bij welke requirements zijn verwerkt in welke functionaliteit(en). Hiervoor zouden in de verschillende onderdelen van functionele ontwerpen verwijzingen naar requirements gemaakt kunnen worden en als metadata in een functioneel ontwerp kunnen worden opgenomen.
- Verwijzingen naar actiepunten (en andere projectinformatie) zouden ook als metadata kunnen worden opgenomen. Hiervoor moet wel een extra waarde worden opgenomen voor ElementType (zie figuur 6.4).
- Acceptatietest-checker: per informatiefunctie kan de status bijgehouden worden. In eerste instantie zal de status iets zijn als 'te ontwerpen' daarna 'in technisch ontwerp', 'in bouw', 'geïmplementeerd', 'goedgekeurd', 'afgekeurd', etc.. Elke informatiefunctie maakt dus onderdeel uit van een workflow. De klant kan zien wat de status is van elke informatiefunctie. Hiervoor dient wel in de InformatiefunctieParagraaf (zie figuur 3.12) een extra veld opgenomen te worden dat de status van de InformatiefunctieParagraaf bijhoudt.

7. Evaluatie

Dit hoofdstuk beschrijft de bevindingen die in dit onderzoek naar voren zijn gekomen. De conclusie beantwoordt in een overzicht de onderzoeksvragen uit hoofdstuk 2. Daarna staat een aantal aanbevelingen op basis van het praktijkonderzoek die niet direct met de functioneel ontwerpdocumentatie te maken hebben. Tot slot worden er mogelijkheden voor verder onderzoek beschreven.

7.1. Conclusie

Het onderzoek ving aan met de vraag hoe Quinity op een goede manier om kan gaan met de toenemende omvang van haar functioneel ontwerpdocumentatie. Om te beginnen is daarvoor in hoofdstuk 3 eerst beschreven welke methode Quinity hanteert voor haar functioneel ontwerpdocumentatie. Dit is gedaan door de ontwerpdocumentatie van Quinity te beschrijven in een UML-model. Het geeft daarmee een overzicht van de onderdelen die in een functioneel ontwerp kunnen voorkomen en hoe die onderdelen met elkaar samenhangen. De kern van het model zijn de verschillende paragraaftypen die verwijzingen naar paragrafen of producten van functioneel ontwerp kunnen bevatten. Ook is beschreven welke onderdelen voor welke doelgroepen bestemd zijn.

Omdat duidelijk moest worden of de functioneel ontwerpdocumentatie aan de behoeften van de betrokkenen voldoet, is er een praktijkonderzoek uitgevoerd. Dit is beschreven in hoofdstuk 4. Het betrof het interviewen van functioneel ontwerpers en technisch ontwerpers en het doornemen van documenten. Ook is gesproken met een aantal klantmedewerkers. Vanuit dit praktijkonderzoek kwamen een reeks eisen waaraan functioneel ontwerpdocumentatie moet voldoen naar voren. Deze eisen vormen de basis voor het voorstel voor vernieuwde functioneel ontwerpdocumentatie dat is beschreven in hoofdstuk 6.

Aan de hand van een overzicht van drie verschillende systeemontwikkelingsmethoden is beschreven wat er in die methoden aan functioneel ontwerp wordt gedocumenteerd. De wijze waarop Quinity haar functioneel ontwerp documenteert is vergelijkbaar met die van SDM / LAD. Deze methode kan daarin gekarakteriseerd worden als een document-gerichte werkwijze. DSDM en RUP gebruiken prototypes en modellen om de functionaliteit te documenteren. Deze prototypes en modellen helpen om het overzicht te bewaren in de functioneel ontwerpdocumentatie.

De belangrijkste uitkomst van het onderzoek is de indicatie dat functioneel ontwerpdocumentatie opgeslagen zou moeten worden volgens het functioneel ontwerpmodel zoals beschreven in paragraaf 3.3 en de aanvullingen daarop in paragraaf 6.1. Met een functioneel ontwerp dat is vastgelegd volgens de regels van het functioneel ontwerpmodel. Voor vastlegging van het functioneel ontwerp volgens dat model moet een functioneel ontwerper wel de juiste gereedschappen hebben waarmee hij een functioneel ontwerp volgens de regels van het model kan

vastleggen. Wanneer dat gereedschap er is, kunnen de eisen die zijn beschreven in paragraaf 4.2 worden behaald .

7.2. Aanbevelingen

Naast het meer modelmatig aanpakken van functioneel ontwerpdocumentatie, zijn er nog een aantal andere verbeterpunten:

- Er zouden standaarden en richtlijnen voor het beschrijven van externe interfacebeschrijvingen moeten komen. De consistentie van externe interfacebeschrijvingen zal daardoor toenemen. Ook is het voor ontwerpers duidelijker welke onderwerpen er in de externe interfacebeschrijvingen moeten staan, en welke onderwerpen daar beter uit weggelaten kunnen worden. Het opstellen van standaarden en richtlijnen zou eenvoudig te bereiken moeten zijn, aangezien er al veel externe interfacebeschrijvingen bestaan. Daar zijn ongetwijfeld punten uit te halen die in iedere externe interfacebeschrijving moeten terugkomen.
- Om verwijzingen te kunnen maken tussen requirements en functioneel ontwerp en tussen functioneel ontwerp en technisch ontwerp, verdient het de aanbeveling om ook de requirementsdocumentatie en technisch ontwerpdocumentatie te modelleren. In dat model kan dan aangegeven worden welke onderdelen in die documentatie geadresseerd kunnen worden. De 'traceerbaarheids-verwijzingen' kunnen dan tussen de requirements, functioneel ontwerp en technisch ontwerp aangegeven worden.

7.3. Verder onderzoek

Verder onderzoek kan in ieder geval gedaan worden naar een concrete implementatie van het voorstel voor vernieuwde functioneel ontwerpdocumentatie. Het eerste dat daarvoor nodig is, is een hulpmiddel waarmee functioneel ontwerp vastgelegd kan worden volgens de structuur van het functioneel ontwerpmodel. Dat model vormt het datamodel van de functioneel ontwerpdocumentatie. Vervolgens kunnen andere hulpmiddelen worden ontwikkeld die gebruik kunnen maken van de structuur van de vernieuwde ontwerpdocumentatie. Een aantal ideeën hiervoor zijn beschreven in paragraaf 6.3.

Het voorstel voor de vernieuwde functioneel ontwerpdocumentatie doet al snel denken aan een opslag in een database, of opslag van documenten in XML. Met XSLT kunnen dan andere documentformaten worden gegenereerd zoals PDF, MS Word, ODF, XHTML, DocBook. Er zou verder onderzoek gedaan kunnen worden hoe het voorstel geïmplementeerd kan worden in één of meer gereedschappen die ontwerpers helpen bij het vastleggen van een functioneel ontwerp in de structuur die ik heb voorgesteld.

Een mogelijke uitbreiding op het functioneel ontwerp is aangeven welke onderdelen daarvan voor welke doelgroepen bestemd zijn en welke niet. Bijvoorbeeld: technische notities zijn alleen bedoeld voor technisch ontwerpers (eventueel ook voor functioneel ontwerpers, maar zeker niet voor de klant). Of ontwerpbeslissingen documenteren, maar dan wel zodanig dat het niet in de

weg staat als je uit een functioneel ontwerp gewoon wilt weten hoe de applicatie werkt (en niet waarom de applicatie zo werkt). Met de juiste gereedschappen kunnen de onderdelen die alleen voor een bepaalde doelgroep van belang zijn, dan ook alleen voor die groep getoond worden. Onderdelen die niet voor hen van belang zijn, worden automatisch niet getoond.

Om het functioneel ontwerpmodel te finetunen, zouden er een paar bestaande functionele ontwerpen in het functioneel ontwerpmodel uitgedrukt kunnen worden. Bij het opstellen van het model is gebruik gemaakt van een aantal functionele ontwerpen en de standaarden en richtlijnen voor functioneel ontwerp. Gedurende het onderzoek zijn er echter geen volledige functionele ontwerpen uitgedrukt in het functioneel ontwerpmodel. Hiermee zou gecontroleerd kunnen worden of er meerdere functionele ontwerpen in het functioneel ontwerpmodel uitgedrukt kunnen worden. Door dit wel te doen, kan het zijn dat er dingen boven tafel komen die nog moeten worden aangepast of toegevoegd aan het model.

Aangehaalde bronnen

D.B. Baarda, M.P.M. de Goede, en J. Teunissen. *Basisboek kwalitatief onderzoek: handleiding voor het opzetten en uitvoeren van kwalitatief onderzoek*. Stenfert Kroese, 2005.

G. Booch, J. Rumbaugh, en I. Jacobson. *The Unified Modeling Language User Guide Second Edition*. Addison-Wesley, 2005.

Y. M. Bosman. *Incorporating functional design patterns in software development*. Masterscriptie, Universiteit Twente, Augustus 2007.

M. Broy, F. Deissenboeck, en M. Pizka. *Demystifying maintainability*. In *WoSQ '06: Proceedings of the 2006 international workshop on Software quality*, pagina's 21–26. ACM, 2006.

Peter Buneman. *Semistructured data*. In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pagina's 117–121. ACM, 1997.

J. Dick. *Design Traceability*. *IEEE Software*, 22:14–16, 2005.

DSDM-Consortium. *DSDM Manual versie 4.2*, 2008. URL `http://www.dsdm.nl/version4/2/members/default.asp`.

L. Fokkinga, M.H. Glastra, en H. Huizinga. *LAD Het lineair ontwikkelen van informatiesystemen*. Academic Service, 2002.

A. Forward en T.C. Lethbridge. *The relevance of software documentation, tools and technologies: a survey*. In *DocEng '02: Proceedings of the 2002 ACM symposium on Document engineering*, pagina's 26–33. ACM, 2002.

R. Guitink en D. Bron. *Functioneel Ontwerp - SDE Standards and Guidelines v1.5.1*. Intern document, augustus 2008.

R. Guitink, D. Bron, en G. Sanders. *Gegevensmodel Standaarden en richtlijnen*. Intern document, Mei 2008.

G. Hodge. *Understanding metadata*. NISO Press, 2004.

S.J.B.A. Hoppenbrouwers, H.A. Proper, en V.E. van Reijswoud. *Navigating the Methodology Jungle – The communicative role of modelling techniques in information system development*. *Computing Letters*, 1(3):97–106, 2005.

J. Kleerekoper. *Design of a Pattern Definition Language*. Masterscriptie, Utrecht University, november 2007.

P. Kruchten. *The Rational Unified Process: an introduction*. Addison-Wesley, 3e editie, 2003.

A. Marcus, X. Xie, en D. Poshyvanyk. *When and how to visualize traceability links?* In *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering*, pagina's 56–61, 2005.

M. Mulders. *Performance Patterns in practice*. Masterscriptie, Erasmus Universiteit Rotterdam, juli 2008.

Peter Nagel. *Functional design patterns: structured reuse of functionality*. Masterscriptie, Universiteit Utrecht, augustus 2006.

R. Neering, R. Batenburg, E. Bunk, F. Harmsen, en S. Brinkkemper. *Het IT-methodenlandschap in Nederland anno 2005. Via selectie, implementatie en toepassing naar succes?* Technisch rapport, Universiteit Utrecht, september 2005.

H.A. Proper, A.A. Verrijn-Stuart, en S.J.B.A. Hoppenbrouwers. *On Utility-based Selection of Architecture-Modelling Concepts*. In S. Hartmann en M. Stumptner, editors, *Second Asia-Pacific Conference on Conceptual Modelling (APCCM2005)*, volume 43 van *CRPIT*, pagina's 25–34. ACS, 2005.

B. Ramesh. *Factors influencing requirements traceability practice*. *Communications of the ACM*, 41:37–44, 1998.

J. Rumbaugh, I. Jacobson, en G. Booch. *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2004.

J. Snijders. *Functional Design Patterns*. Masterscriptie, Utrecht University, augustus 2004.

G. Spanoudakis. *Plausible and adaptive requirement traceability structures*. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pagina's 135–142. ACM, 2002.

W.T. van de Molengraft. *Performance patterns for relational databases*. Masterscriptie, Technische Universiteit Eindhoven, augustus 2007.

J. van Helden en N. Reyngoud. *Functional Design Patterns*. Masterscriptie, Universiteit Utrecht, december 2005.

J.J.E. van Montfort. *Functional Design Patterns: De implementatie van model-gedreven functionaliteit in een object georišenteerde omgeving.* Masterscriptie, Technische Universiteit Eindhoven, augustus 2006.

K.J. van Steenbergen. *User Interface Design Patterns for Rich Internet Applications*. Masterscriptie, Technische Universiteit Eindhoven, augustus 2007.

G. Vlasblom, D.B.B. Rijsenbrij, en G.P.A.J. Delen. *Produkten van systeemontwikkeling*. Cap Gemini Publishing, 1991.

A. Woordenlijst

Hieronder is een overzicht met belangrijke woorden / termen / begrippen die ik gebruik.

Functioneel ontwerp Een beschrijving van gegevens in een informatiesysteem, gedrag van een informatiesysteem en de manier waarop gebruikers kunnen interacteren met een informatiesysteem. Deze beschrijving is zodanig dat klantmedewerkers kunnen controleren wat de klant krijgt en dat ontwikkelaars het informatiesysteem kunnen implementeren.

Informatiefunctie Een (door mens of computer) verrichte functie binnen het kader van een informatiesysteem. Hiermee wordt bedoeld: een afgebakend en samenhangend geheel van activiteiten voor het produceren, verwerken of beheren van gegevens in dienst van het doel van het informatiesysteem. Bij het verdelen van een informatiesysteem in informatiefuncties, moeten deze in ieder geval zó laag worden gekozen dat er binnen een informatiefunctie sprake is van eenheid in tijd, plaats en afhandeling. Binnen die beperking moeten de informatiefuncties zo hoog mogelijk gekozen worden (Fokkinga et al., 2002).

Informatiesysteem Een stelsel van gegevens, informatiesysteemfuncties en eventueel systeemgebonden technische structuur. Het doel ervan is: het verzamelen, vastleggen, verwerken, bewaren, transporteren en verstrekken van gegevens ten behoeve van het nemen van beslissingen (door mensen, programma's of apparatuur) op een afgebakend toepassingsgebied of ten behoeve van een bepaald bedrijfsdoel (Fokkinga et al., 2002).

Klant Een persoon of groep personen die de eisen en wensen over het informatiesysteem specificeert.

Requirement Een eis of wens van de klant over het te bouwen systeem. Er is onderscheid tussen *functionele* en *non-functionele* requirements. Een functionele requirement is een eis of wens over de functionaliteit van het systeem. Een non-functionele requirement is een eis of wens over de randvoorwaarden waarmee het systeem om moet gaan.

Systeemontwikkeling Het proces waardoor een Informatiesysteem tot stand komt. Vaak is dit proces opgedeeld in een aantal fasen.

Technisch ontwerp Een beschrijving van de technische ontwerpkeuzes en de wijze waarop de functionaliteit uit het functioneel ontwerp is geïmplementeerd in de code. Een functioneel ontwerp kan technisch op meerdere manieren worden geïmplementeerd. Een technisch ontwerp is een verantwoording en een beschrijving van die keuzes.

Use Case Een beschrijving van de stappen die het systeem uitvoert om een bepaald doel voor een actor te bereiken.

B. Lijst van afkortingen

DSDM Dynamic Systems Development Method

ER Entity Relationship

FO Functioneel ontwerp

HTML HyperText Markup Language

LAD Linear Application Development

ORM Object Role Modeling

QIS Quinity Insurance Solution

RAD Rapid Application Development

RUP Rational Unified Process

SDM System Development Methodology

TO Technisch ontwerp

UML Unified Modeling Language

C. Vragen interviews functioneel en technisch ontwerpers

(Cursief gedrukte vragen zijn beschrijvend van aard)

C.1. Vragen interview functioneel ontwerpers

De uitwerking en analyse van dit interview is te vinden in bijlage A en B van het document 'Vertrouwelijke bijlagen Behorende bij de masterscriptie "Gestructureerde documentatie van functioneel ontwerp"'.

Ik heb de volgende vragen aan functioneel ontwerpers voorgelegd:

- *Wat versta je onder functioneel ontwerp, hoe definieer je dat?*
- *Welke onderdelen onderscheid je in een functioneel ontwerp?*
- *Voor wie maak je een functioneel ontwerp? Welke doelgroep heb je voor ogen als je een functioneel ontwerp opstelt?*
  - *Maak je daarbij onderscheid tussen de verschillende onderdelen van het functioneel ontwerp (functionele beschrijving, externe interfaces, gegevensmodel, prototype)?*
- *Houd je rekening met de technische achtergrond van de klant als je een functioneel ontwerp opstelt? De ene klant zal bijvoorbeeld technischer ingesteld zijn dan een andere klant*
  - *Kun je voorbeelden geven waar je dan rekening mee houdt en op welke manier je dat doet?*
  - *Is er hierbij onderscheid tussen de verschillende onderdelen van het functioneel ontwerp (functionele beschrijving, externe interfaces, gegevensmodel, prototype)?*
- Ken je de standaarden en richtlijnen voor het functioneel ontwerp en gebruik je ze ook in je dagelijks werk?
  - Waarom wel of niet? Welke onderdelen gebruik je?
- Heb je 'last' van bepaalde richtlijnen?
  - Zo ja, welke dan, en waarom?
- Is het voor een functioneel ontwerper van belang dat je stukken functioneel ontwerp op de requirements kunt terugvoeren?

  - *Hoe gebeurt dat op dit moment?*
  - *Is er hierbij onderscheid tussen de verschillende onderdelen van het functioneel ontwerp (functionele beschrijving, externe interfaces, gegevensmodel, prototype)?*

C.2. Vragen interview technisch ontwerpers

De uitwerking en analyse van dit interview is te vinden in bijlage C en D van het document 'Vertrouwelijke bijlagen Behorende bij de masterscriptie "Gestructureerde documentatie van functioneel ontwerp"'.

Technisch ontwerpers heb ik voor een deel dezelfde vragen voorgelegd. Daarnaast ook een aantal vragen die specifiek op hun situatie ingaan.

- *Wat versta je onder functioneel ontwerp, hoe definieer je dat? Wat is het verschil met technisch ontwerp?*
- *Welke onderdelen uit het functioneel ontwerp gebruik je bij het opstellen van het technisch ontwerp?*
- *Op basis van welke informatie uit het functioneel ontwerp maken technisch ontwerpers het datamodel?*
- Zijn er onderdelen in het functioneel ontwerp die voor jou als technisch ontwerper **niet** van belang zijn?
  - *Welke onderdelen zijn dat?*
- *Welke onderdelen van het functioneel ontwerp bevatten vaak fouten?*
- Wat voor soort informatie mis je regelmatig in een functioneel ontwerp?
- Hoe goed of hoe slecht vind je de structuur van de functionele beschrijvingen? Kun je terugvinden wat je zoekt?
- Hoe makkelijk of hoe moeilijk zijn de verschillende onderdelen van het functioneel ontwerp (algoritmes, schermen, schermflow, workflow) over het algemeen naar het technisch ontwerp om te zetten? Is er duidelijk aangegeven welke gegevens je daarvoor nodig hebt?
- Hoe duidelijk of hoe onduidelijk zijn de algoritmes over het algemeen gespecificeerd? Is de specificatie voldoende om het algoritme te kunnen implementeren?
- Voegen de illustraties uit het functioneel ontwerp wat toe om het technisch ontwerp op te stellen?
- Is het voor een technisch ontwerper van belang dat je stukken code op informatiefuncties kunt terugvoeren?
  - *Hoe gebeurt dat op dit moment?*