Onderzoeksvoorstel

Dennis Janse 0213861
<d.janse@student.science.ru.nl>
<dennis.janse@quinity.com>

5 maart 2008

1 Inleiding

Dit onderzoeksvoorstel heeft als doel verantwoording af te leggen aan Quinity en de universiteit waarom het onderzoek gedaan wordt, wat het onderzoek inhoudt en hoe het onderzoek uitgevoerd wordt. Het doel van het onderzoek is een taal te ontwikkelen die het functioneel ontwerp op een gestructureerde wijze kan beschrijven. Hierbij moet rekening gehouden worden met de begrijpelijkheid, herbruikbaarheid en onderhoudbaarheid van functionele ontwerpen.

2 Context van het onderzoek

Quinity maakt e-business software voor met name de financiële sector. De applicaties die het bedrijf levert, worden op maat gemaakt op basis van standaardcomponenten.

In het ontwikkelproces van een concrete applicatie is het ontwerp van de applicatie een belangrijk onderdeel. Eerst wordt een functioneel ontwerp opgesteld dat specificeert wat het systeem kan 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 samenhang tussen de verschillende onderdelen weergeeft en een prototype dat alle schermen 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.

Om niet telkens opnieuw het wiel te hoeven uitvinden is het handig om eerdere vindingen te hergebruiken. Ook kunnen nieuwe applicaties dan sneller en goedkoper worden ontwikkeld. De kwaliteit van die applicaties kan ook worden verhoogd doordat er gebruik gemaakt wordt van "best practices". Dit wordt voor programmacode en technische ontwerpen al geruime tijd gedaan. Quinity is sinds een aantal jaren bezig met onderzoek hoe ook functionele ontwerpen hergebruikt kunnen worden.

Allereerst is het begrip functioneel ontwerppatroon gedefinieerd door Jeroen Snijders. Daarnaast heeft hij laten zien hoe deze patronen gebruikt kunnen worden en hoe deze techniek zich verhoudt tot andere technieken. Vervolgens is deze definitie verder uitgewerkt door Jeffrey van Helden en Niels Reyngoud. Zij hebben een sjabloon gedefinieerd waarin functionele ontwerppatronen kunnen worden gedocumenteerd. Jeroen van Montfort heeft onderzocht hoe functionele ontwerppatronen gebruikt kunnen worden voor het maken van technische ontwerpen en implementatie van code. De laatste onderzoeken naar functionele ontwerppatronen zijn gedaan

door Jacob Kleerekoper en Yoran Maxim Bosman. Kleerekoper stelt een een taal voor waarmee functionele ontwerppatronen beschreven kunnen worden. Dit omvat de documentatiestructuur van functionele ontwerppatronen, gebruikte kernbegrippen en verschillende typen diagrammen die op te bouwen zijn met representaties van die kernbegrippen. In dit onderzoeksvoorstel zal ik deze taal verder PDL[1] noemen. Bosman beschrijft hoe functionele ontwerppatronen in het functioneel ontwerp en functioneel ontwerpproces kunnen worden gebruikt.

3 Conceptueel model

De belangrijkste begrippen binnen dit onderzoek en de verbanden tussen deze begrippen wil ik in deze paragraaf bespreken. Gedurende het onderzoek zal dan helder zijn waar het over gaat als deze begrippen gebruikt worden. Wanneer ik een begrip definieer, zal ik deze in **Dikgedrukte** tekst en een hoofdletter weergeven.

Het begrip waarin alle andere belangrijke begrippen binnen dit onderzoek zijn in te bedden is **Systeemontwikkeling**. Dit is het proces waardoor het te bouwen Informatiesysteem tot stand komt. Met een **Informatiesysteem**[2] bedoel ik een computergebaseerd "systeem voor informatieverwerking en -overdracht" [van Dale, 2008]. In de literatuur zijn veel verschillende manieren te vinden waarop je het Systeemontwikkelingsproces kunt inrichten. Bij Quinity zijn de systeemontwikkelingsprojecten gebaseerd op Linear Application Development (LAD) en Dynamic Systems Development Method (DSDM). In het algemeen bestaan deze projecten uit de volgende fasen:

- Requirementsfase;
- Functioneel ontwerpfase;
- Realisatiefase, met daarin de subfasen:
  - Technisch Ontwerpfase;
  - Constructiefase;
- Acceptatietestfase;
- Invoeringsfase.

Voor sommige projecten zijn er ook nog fasen die parallel lopen aan deze fasen, maar die neem ik hier verder niet in beschouwing[3]. In figuur 1 is de fasering van het systeemontwikkelingsproces grafisch weergegeven.

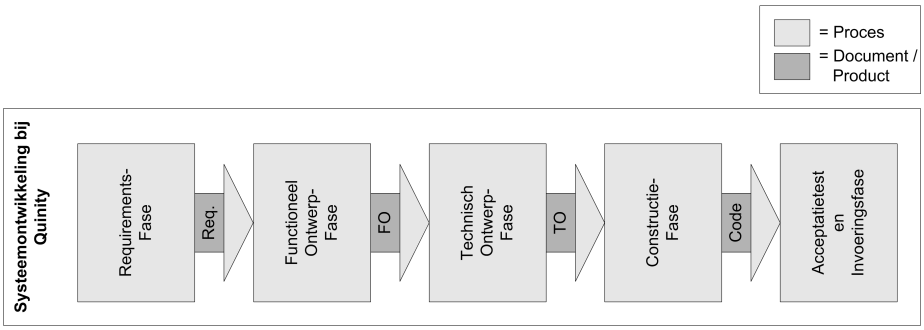
3.1 Requirementsfase

Een **Requirement** is een eis of wens van de klant over het te bouwen systeem. Er wordt onderscheid gemaakt in functionele en niet-functionele Requirements. Een functionele Requirement is een Requirement die de functionaliteit van het systeem beschrijft. Een niet-functionele Requirement is een Requirement die beperkingen

[1] Pattern Definition Language

[2] Het woord Informatiesysteem kort ik vaak af tot 'systeem'. Uit de context zal blijken als ik het heb over een ander soort systeem.

[3] Dit zijn fasen als opleidingen, organisatorische inrichting en gegevensconversie.



Figuur 1: Systeemontwikkelingsfasen

of randvoorwaarden beschrijft waarmee het systeem om moet gaan. De Requirementsfase is bedoeld om de Requirements te vinden. Aan het einde van deze fase is er een document waarin de Requirements beschreven zijn. Dit is de invoer voor de Functioneel Ontwerpfase, waarin het Functioneel Ontwerp wordt opgesteld.

3.2 Functioneel Ontwerpfase

Een **Functioneel Ontwerp** is een beschrijving van de wijze waarop het systeem de gewenste functionaliteiten realiseert. Het Functioneel Ontwerp moet daarvoor zo beschreven zijn dat het:

- voor de klant duidelijk is wat Quinity gaat realiseren en
- voor de programmeur duidelijk is wat hij moet bouwen.

Het ontwerp bestaat uit vier delen:

1. **Functionele Beschrijving** - een beschrijving van alle functies binnen het systeem: de transformatie tussen invoer en uitvoer van gegevens;
2. **Datamodel** - een beschrijving van de gegevensgroepen en hoe het systeem deze opslaat;
3. **Prototype** - een beschrijving van de schermen waardoor de gebruiker kan interacteren met het systeem;
4. **Externe interfaces** - een beschrijving van de uit te wisselen informatie met externe systemen en welke procedures hiervoor nodig zijn.

Al deze delen worden met de klant gecommuniceerd. Het grootste gedeelte van de Functionele Beschrijving bestaat uit een beschrijving van de Informatiefuncties. De beschrijving is in natuurlijke taal, met illustraties ter verduidelijking. Wanneer een bedrijfsproces wordt beschreven, gaat dat in termen van de Informatiefuncties die een gebruiker nodig heeft om dat proces te ondersteunen. Een **Informatiefunctie** is een door een mens (handmatig) of computer (geautomatiseerd) verrichte functie binnen een informatiesysteem. Binnen de functie is sprake van eenheid van tijd, plaats en handeling. Een beschrijving van een Informatiefunctie geeft weer wat de invoer is van de functie, wat de functie doet en wat de uitvoer is van de functie.

Binnen het Functioneel Ontwerp kan er gebruik gemaakt worden van **Functionele Ontwerppatronen**. Dit zijn patronen die terugkerende functionele aspecten van (domeinspecifieke) applicaties beschrijven [Snijders, 2004, p. 21]. Deze patronen zijn opgesteld aan de hand van eerdere Functionele Ontwerpen. Het Functioneel Ontwerppatroon zorgt dus voor hergebruik op functioneel niveau. Dit type patroon kan beschreven worden met PDL. Deze Taal is een **Metataal**, een taal waarmee andere talen beschreven kunnen worden . PDL legt vast welke constructen er zijn en hoe deze constructen gebruikt mogen worden om functionaliteit te beschrijven. **Taal** is een vehikel om tussen twee of meer personen te kunnen communiceren zodat de zender aan de ontvanger(s) kan duidelijk maken wat hij bedoelt. In deze zin hoeft taal dus niet slechts een formalisme zoals een programmeertaal of logica te zijn. De taal kan meerdere onderdelen omvatten, zoals een diagramtechniek, structuur van een functioneel ontwerpdocument, structuur van het functioneel ontwerpproces; alles wat er aan bijdraagt om de boodschap van de zender zo goed mogelijk over te brengen naar de ontvanger.

3.3 Technisch Ontwerpfase

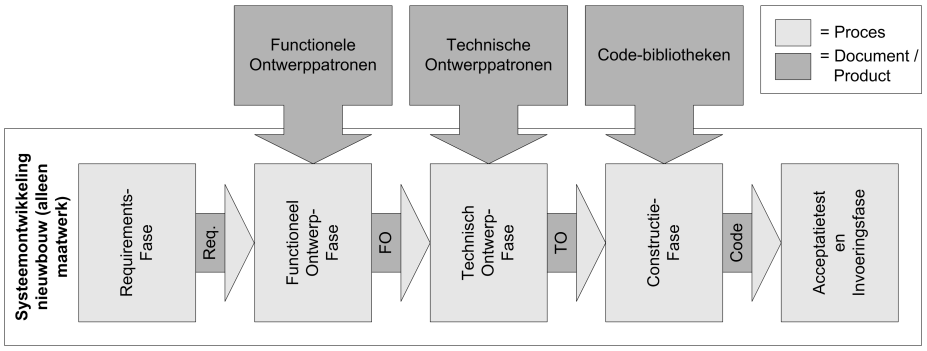
De Technisch Ontwerpfase is de derde fase in de systeemontwikkeling. Het **Technisch Ontwerp** is een beschrijving over de wijze waarop de functionaliteit uit het Functioneel Ontwerp moeten worden geïmplementeerd. Verschillende diagrammen beschrijven daarin hoe de verschillende klassen met elkaar samenhangen en welke objecten de gewenste functionaliteit implementeren. Ten opzichte van het Functioneel Ontwerp is het Technisch Ontwerp gericht op het systeem in plaats van de gebruiker. Bij dit ontwerp kan gebruik gemaakt worden van **Technische Ontwerppatronen**. Dit zijn beschrijvingen van communicerende objecten en klassen om een algemeen ontwerpprobleem in een bepaalde **context** op te lossen [Gamma et al., 1999, p. 3]. Het Technisch Ontwerppatroon zorgt dus voor hergebruik op technisch niveau.

3.4 Constructiefase

In de constructiefase wordt de code gemaakt. Het resultaat is (gecompileerde) programmacode die het Technisch Ontwerp in iets werkends realiseren. Ook hier speelt hergebruik een rol door het gebruik van **Code-bibliotheken**. Dit zijn verzamelingen van klasses die **algemene** functionaliteit implementeren. Het hergebruik in deze fase is dus eenvoudiger te realiseren dan in het Technisch Ontwerp. Daar moet namelijk het patroon worden toegepast op de context van een concreet ontwerp. Het vereist een vertaling van het patroon naar een bepaald ontwerp. In de constructiefase kan bestaande code eenvoudig worden aangeroepen vanuit de nieuw te bouwen code, er is geen aanpassing van de hergebruikte code nodig.

3.5 Acceptatietest en invoeringsfase

In de voorlaatste stap wordt het gebouwde systeem door de klant getest of het aan zijn eisen voldoet. Als het systeem niet wordt geaccepteerd, dienen de fouten die hiervan de oorzaak zijn opgelost te worden. Afhankelijk van de soorten fouten moeten hiervoor één of meer voorgaande stappen worden herhaald. Na acceptatie volgt de daadwerkelijke invoering van het ontwikkelde systeem. Dit omvat de koppeling



Figuur 2: Conceptueel model nieuwbouw

met andere systemen, begeleiding van eindgebruikers en, indien van toepassing, de conversie van het oude systeem naar het nieuwe systeem.

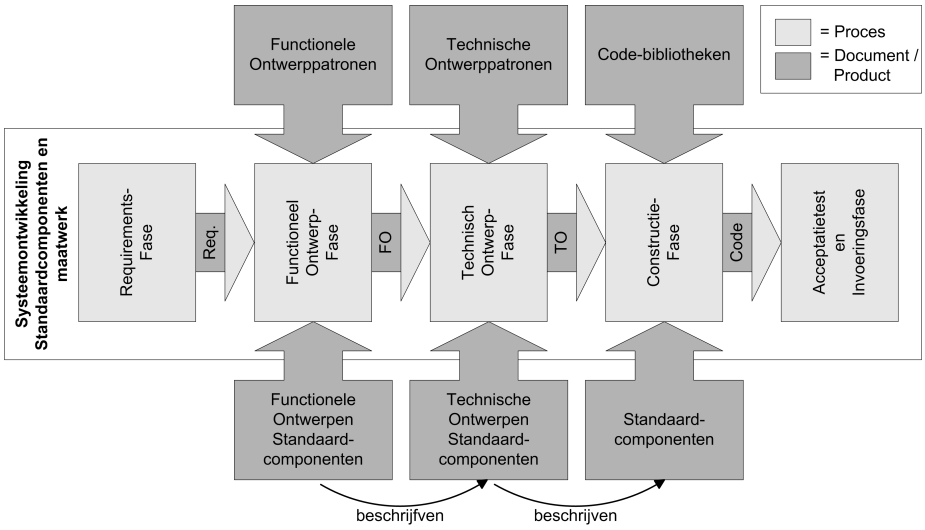
In figuur 2 is het systeemontwikkelingsproces nogmaals grafisch weergegeven. Nu zijn ook de onderdelen die met hergebruik te maken hebben opgenomen. De drie niveaus van hergebruik zijn in de figuur terug te zien. Door Functionele Ontwerpen, Technische Ontwerpen en Code te hergebruiken, kan de ontwikkeling van toekomstige systemen tegen lagere kosten, in kortere tijd en met hogere kwaliteit gedaan worden.

Elke stap in het proces gebruikt één of meerdere bronnen als invoer en heeft één of meerdere producten als resultaat. Het Functioneel Ontwerpproces gebruikt dus bijvoorbeeld de Requirements en Functionele Ontwerppatronen en levert het Functioneel Ontwerp als resultaat. Andere invoerbronnen in het Functioneel Ontwerpproces zijn bijvoorbeeld notulen van overlegsessies met de klant en standaarden en richtlijnen met betrekking tot het Functioneel Ontwerp.

Nu geldt dit model in principe voor alle projecten binnen Quinity. Moet er echter een systeem ontwikkeld worden op basis van Standaardcomponenten, dan komen er nog een paar documenten bij. Een deel van de functionaliteit is dan al gerealiseerd in de Standaardcomponenten, een ander deel is maatwerk-systeemontwikkeling. Een **Standaardcomponent** is een op zichzelf staand systeem dat functionaliteit levert voor een bepaald domein. Anders dan bij Code-bibliotheken is dit dus een geheel van samenwerkende onderdelen; Code-bibliotheken bieden alleen kleine stukken niet-domeinspecifieke functionaliteit. In figuur 3 is het systeemontwikkelingsproces weergegeven waarbij gebruik wordt gemaakt van Standaardcomponenten. Omdat de functionaliteit die de Standaardcomponent biedt ook gedocumenteerd moet worden in het Functioneel Ontwerp, moet het Functioneel Ontwerp van de Standaardcomponent, of in ieder geval het relevante deel daarvan, ook in het Functioneel Ontwerp van het te bouwen systeem komen te staan. Er is ook een Technisch Ontwerp voor de Standaardcomponent. In de constructiefase wordt natuurlijk ook de Standaardcomponent zelf gebruikt.

4 Doelstelling / vraagstelling

Zoals Kleerekoper een taal heeft ontworpen om functionele ontwerppatronen te kunnen beschrijven [Kleerekoper, 2007], zo zou er ook een taal moeten zijn om



Figuur 3: Conceptueel model maatwerk op basis van standaardcomponenten

functionele ontwerpen mee te beschrijven. Op dit moment zijn er binnen Quinity standaarden en richtlijnen voor het functioneel ontwerp en het ontwerpproces. Deze helpen om het functioneel ontwerp te communiceren tussen klant en technisch ontwerper. Quinity geeft echter aan dat deze standaarden en richtlijnen te weinig houvast bieden om het functioneel ontwerp op te stellen. Dat komt terug op vier vlakken:

- Integratie van functionele ontwerpen van standaardcomponenten in functionele ontwerpen van projecten;
- Functionele beschrijvingen in het functioneel ontwerp zijn niet altijd duidelijk genoeg voor de klanten;
- Functioneel en technisch ontwerpers moeten omgaan met grote hoeveelheden documentatie rond het ontwerp;
- Integratie van functionele ontwerppatronen in functionele ontwerpen.

Door het gebruik van de te ontwikkelen taal moet het functioneel ontwerp op een gestructureerde wijze worden opgesteld. Het doel daarbij is dat functionele ontwerpen zodanig verbeterd worden dat de klant beter weet wat Quinity gaat maken. Deze taal moet eenvoudig te begrijpen zijn voor de klant aangezien deze niet dezelfde technische kennis bezit als de medewerkers van Quinity. Ook moeten technisch en functioneel ontwerpers inzicht krijgen en houden in de werking van de standaardcomponenten, wat moeilijk kan zijn door de omvang van de documentatie daarvan. Daarnaast zou het handig zijn als PDL kan worden ingezet voor functionele ontwerpen. Functionele ontwerppatronen die beschreven zijn met PDL kunnen dan waarschijnlijk eenvoudiger worden gebruikt in functionele ontwerpen.

De hoofdvraag is dus wat een geschikte taal is voor het beschrijven van functionele ontwerpen. Hoe ziet deze taal er uit en aan welke eisen moet deze taal voldoen?

- **Vraag 1:** Wat is een geschikte taal voor het beschrijven van functionele ontwerpen?
  - Welke knelpunten zijn er nu bij het opstellen van het functioneel ontwerp?
  - Aan welke eisen moet deze taal voldoen?
  - Kan PDL in deze taal gebruikt worden?

Omdat nog niet duidelijk is wat de eisen voor de te ontwikkelen taal zijn, moet eerst onderzocht worden wat de eisen hiervoor zijn. Een eerste stap hierin is onderzoek naar **wat** de onderstaande drie punten beschrijven (welke concepten) en **hoe** deze punten tot stand komen (welke processen):

1. functionele ontwerpen binnen Quinity;
2. functionele ontwerpen zoals de literatuur hier standaarden en richtlijnen voor geeft;
3. functionele ontwerppatronen, opgesteld met behulp van PDL.

Op basis van het eerste punt kan ik aangeven welke aspecten er nu in het functioneel ontwerp voorkomen. Het tweede punt zal inzicht geven in welke aspecten er belangrijk zijn volgens de literatuur. Een vergelijking tussen deze twee zal leiden tot een overzicht van aspecten die wel, niet voldoende of niet voorkomen in het functioneel ontwerp bij Quinity. Ook het proces waardoor een functioneel ontwerp tot stand komt is van belang. Hier kan ook een vergelijking worden gemaakt van het proces binnen Quinity en processen zoals de literatuur daar standaarden en richtlijnen voor geeft. Dit zal leiden tot inzicht in welke aspecten van het functioneel ontwerpproces nu wel, niet voldoende of niet aanwezig zijn binnen Quinity. Het derde punt is van belang om er achter te komen welke aspecten PDL beschrijft die gemeenschappelijk zijn met functionele ontwerpen zoals die nu gemaakt worden binnen Quinity.

Vragen 2, 3 en 4 worden dus:

- **Vraag 2a en 2b:** Wat wordt er beschreven in functionele ontwerpen binnen Quinity? Hoe worden deze ontwerpen opgesteld?
- **Vraag 3a en 3b:** Wat moet er volgens de literatuur beschreven worden in functionele ontwerpen? Welke standaarden en richtlijnen geeft de literatuur hier voor?
- **Vraag 4:** Wat wordt er beschreven in functionele ontwerppatronen die opgesteld zijn met behulp van PDL?

Naast een literatuuronderzoek naar de aspecten van functionele ontwerpen, is het ook van belang om te weten te komen of de klanten het functioneel ontwerp

goed begrijpen. Hiervoor zal ik een enquête houden die knelpunten in het functioneel ontwerp ten aanzien van de klant in kaart brengt.

Het is van belang dat ook de technische ontwerpers een voldoende specifieke taal hebben. Lopen zij op dit moment misschien ook tegen knelpunten aan in het functioneel ontwerp? Is het functioneel ontwerp wel voldoende specifiek om het technisch ontwerp te kunnen maken?

Ook functioneel ontwerpers hebben belang bij een zo goed mogelijk functioneel ontwerp. Met name waar zij ontwerpen moeten opstellen die gebruik maken van standaardcomponenten. Wat vinden zij van het funtioneel ontwerp? Kunnen zij uit de voeten met het ontwerp zoals dat nu wordt opgesteld?

- **Vraag 5:** Wat vinden de klanten van het functioneel ontwerp?
  - Is het ontwerp duidelijk?
  - Zijn de gebruikte schema's te begrijpen?
- **Vraag 6:** Wat vinden de technisch ontwerpers van het functioneel ontwerp?
  - Is het ontwerp duidelijk genoeg om het technisch ontwerp te maken?
  - Zijn de schema's voldoende specifiek?
- **Vraag 7:** Wat vinden de functioneel ontwerpers van het functioneel ontwerp?
  - Kan het functioneel ontwerp van standaardcomponenten gemakkelijk worden ingebed in functionele ontwerpen van projecten?

Nadat ik dus in kaart heb gebracht wat de gebreken zijn aan het functioneel ontwerp en wat de gebreken zijn in het functioneel ontwerpproces, zijn de eisen voor de taal die ik op zal stellen duidelijk. Daarnaast is duidelijk welke aspecten wel en niet door PDL of een deel van PDL[4] ondersteund kunnen worden. Op basis van deze gegevens zal ik de taal voor het functioneel ontwerp kunnen ontwerpen.

5 Onderzoeksmodel

In deze paragraaf bespreek ik hoe mijn onderzoek is opgezet. Hoe kom ik tot de beantwoording van de vragen en in welke termijn doe ik dat. Verder geef ik aan welke producten worden opgeleverd en wat de vertrouwelijkheid van het onderzoek inhoudt.

5.1 Organisatie van het onderzoek

Ik zal beginnen bij Quinity met een inwerkperiode van een maand. Tijdens die periode kan ik in ieder geval een cursus datamodellering volgen en misschien ook

[4] Bijvoorbeeld kunnen alleen de schema's worden gebruikt.

een cursus functioneel ontwerpen. Daarnaast zal ik een aantal interne documenten doornemen. Een aantal daarvan gaan over standaarden en richtlijnen rond het functioneel ontwerp. Een ander deel zijn functionele ontwerpen van gerealiseerde of nog lopende projecten. Hierdoor zal ik een overzicht krijgen van de kernaspecten van het functioneel ontwerp zoals dat bij Quinity gerealiseerd wordt.

Vervolgens zal ik een literatuuronderzoek doen naar functioneel ontwerp. Dit onderzoek moet de aspecten van functioneel ontwerp ontdekken die van belang zijn om tot een goed ontwerp te komen. De resultaten van dit onderzoek zal ik vergelijken met het functioneel ontwerp zoals dat binnen Quinity wordt opgesteld.

Aan de hand van het literatuuronderzoek kan ik ook een enquête opstellen over de vraag wat de klanten van het functioneel ontwerp vinden. De literatuur kan aanwijzingen geven welke aspecten van belang zijn bij het funtioneel ontwerp en hoe die verschillende aspecten kunnen worden gemeten. In eerste instantie zal ik de enquête afnemen bij één project. Hierdoor krijg ik inzicht in de goede punten en de knelpunten bij dat project. Dit inzicht kan ik gebruiken bij het opstellen van de taal.

Nadat de enquête is afgenomen en verwerkt, kan ik de taal voor het functioneel ontwerp op gaan stellen. Hierbij zal ik moeten kijken welke aspecten door PDL ondersteund kunnen worden, of dat ik een geheel nieuwe taal maak. Ik moet nog bepalen op welke wijze ik de taal ga opstellen.

Om te valideren of het opstellen van de taal goed is gelukt, zal eigenlijk een project uitgevoerd moeten worden waarin die taal is gebruikt. Na afloop van het project kan de enquête dan nogmaals worden afgenomen (over dat project). Aan de hand daarvan kan gecontroleerd worden of er verbetering is. Deze validatie zal ik niet tijdens mijn afstudeerstage kunnen doen, aangezien er te weinig tijd is om een heel systeemontwikkelingsproject binnen mijn afstudeertijd af te ronden. Er moet dus een andere manier komen waarmee ik de te ontwerpen taal kan valideren. Een manier om dat te doen is aan de hand van de vooraf op te stellen eisen. Zo kan ik achteraf controleren of de taal aan die eisen voldoet. Hierbij moet je er wel vanuit gaan dat de eisen die aan de taal gesteld zijn ook het probleem oplossen. Een andere manier is om kleine stukjes functioneel ontwerp in de oude en de nieuwe taal op te stellen. Deze stukjes kun je dan voorleggen aan de mensen die er mee moeten werken. Je kunt dan niet alle onderdelen van de taal toetsen, maar in ieder geval wel een deel daarvan. Het belangrijkste is dat de manier van werken systematisch en gestructureerd is, zodat het resultaat te controleren valt.

5.2 Te gebruiken literatuur

Voor het literatuuronderzoek wil ik een aantal artikelen doornemen over de onderwerpen Component Based Development (CBD), Commercial Off-The-Shelf software (COTS) en software ontwikkelingsmethoden. De eerste twee onderwerpen gaan over het ontwikkelen van software op basis van standaardcomponenten. In Lau en Wang [2007] wordt ingegaan op het gebruik van software componentmodellen. Dit zijn modellen die definiëren wat componenten zijn en hoe die samenhangen. Het artikel geeft een aantal criteria waaraan de modellen zouden moeten volden. Vervolgens worden enkele modellen besproken en aangegeven in hoeverre ze aan de opgestelde eisen voldoen.

Vigder en Dean [1997] schreven een een artikel waarin ervaringen met het integreren van COTS software in bestaande systemen worden beschreven. Aan de hand

van die ervaringen zijn een aantal informele regels opgesteld die de softwareontwikkeling voor dit soort systemen moet vergemakkelijken.

Een aantal artikelen gaan over software ontwikkelingsmethoden. McGregor en Korson [1994] gaat over de integratie van het ontwikkelingsproces met het testproces. Niet alleen testen van code is belangrijk, maar ook testen van tussenliggende documenten, modellen en ontwerpen. Door het testproces moeten de verschillende producten toenemen in kwaliteit. Bhatti [2005] bespreekt een aantal metrieken om kwaliteit te meten van verschillende onderdelen in het ontwikkelingsproces.

Vaak word er een spanning ervaren tussen modellen voor softwareontwikkeling en de praktijk daarvan. Volgens **?** is dit het gevolg van twee verschillende visies op ontwerp. Hij geeft aan wat er voor nodig is om deze spanning te verlagen.

Naast deze artikelen zijn waarschijnlijk ook de boeken van Pressman [2005] en Evans [2004] nuttig om te gebruiken bij het literatuuronderzoek. De eerste gaat over allerlei aspecten van software ontwikkeling, waaronder domeinmodellering, dat veel met functioneel ontwerp te maken heeft. Het tweede boek gaat over hoe domeinmodellering in het software ontwikkelingsproces geïntegreerd kan worden. Onder andere begrippen als "gemeenschappelijke taal" en "model driven design" komen aan bod.

5.3 Planning

In figuur 4 staat de planning van mijn onderzoek weergegeven. De belangrijkste taken zijn daarin aangegeven samen met het tijdpad waarbinnen deze uitgevoerd moeten worden. Zoals te zien ben ik gedurende de hele tijd bezig met het schrijven van mijn scriptie. Dit document zal gedurende het project steeds verder groeien en dient als uitgangspunt voor de gesprekken met de begeleiding vanuit de universiteit.

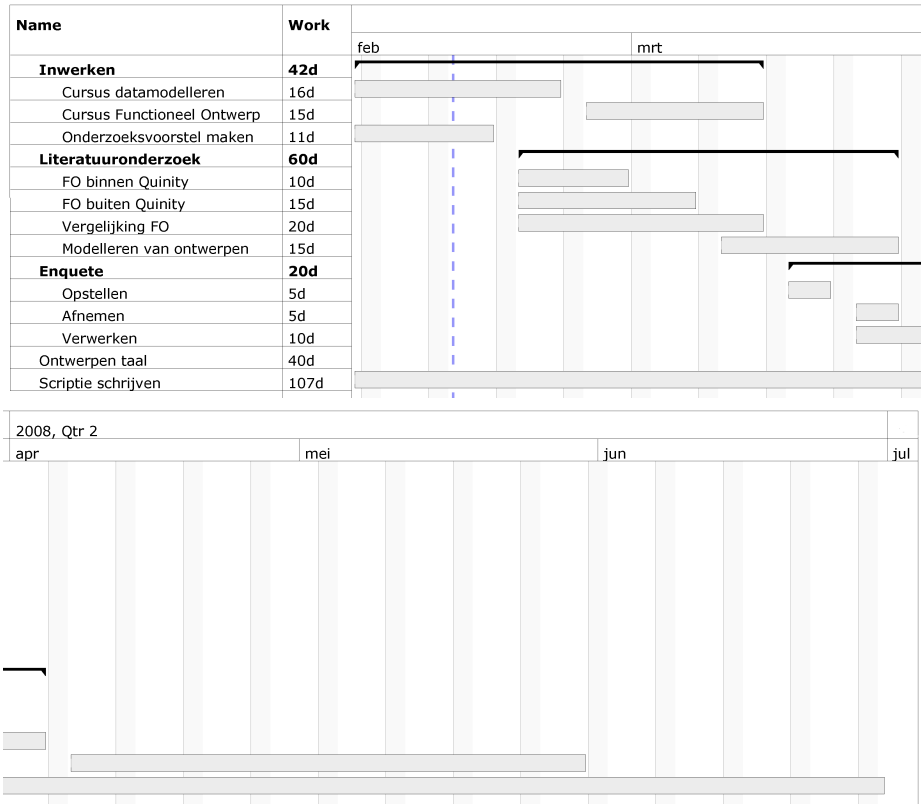
5.4 Producten

Welke producten worden opgeleverd aan de RU en Quinity staat in tabel 1. Het is de bedoeling dat met de begeleider van de RU ongeveer elke maand een update van het onderzoek wordt opgeleverd. Dit omvat drie delen:

1. Een actuele print van het volledige afstudeerdocument;
2. Een puntsgewijze opsomming van de activiteiten in de afgelopen periode op een aparte pagina geprint;
3. Een puntsgewijze opsomming van de geplande activiteiten voor de komende periode op een aparte pagina geprint.

Datum	Product	Aan wie
do. 31-01	Onderzoeksvoorstel	RU, Quinity
do. 14-02	Update onderzoek	RU
do. 13-03	Update onderzoek	RU
...	...	...
ma. 30-06	Scriptie	RU, Quinity

Tabel 1: op te leveren producten



Figuur 4: Planning

5.5 Vertrouwelijkheid

Niet alle documenten bij Quinity zijn publiekelijk in te zien. Hierover is in de afstudeerovereenkomst onder artikelen 10.6 en 10.7 het volgende overeengekomen:

10.6 Al het materiaal dat betrekking heeft op de functionele en technische inhoud van de werkzaamheden is vertrouwelijk. Dit betreft onder meer functioneel ontwerpen, functionele patronen, technische ontwerpen, technische patronen, programmatuur, rapporten en overzichten, etc. Dit materiaal mag niet buiten Quinity gebracht worden zonder schriftelijke toestemming van Quinity.

10.7 Het materiaal dat betrekking heeft op de methodiek en notatiewijze voor het gebruik van functionele ontwerppatronen mag in de afstudeerscriptie gebruikt worden. Dit betreft dus algemene zaken die projectoverstijgend zijn en geen referentie hebben naar specifieke functionaliteit.

Aangehaalde bronnen

Shahid Nazir Bhatti. **Why quality?: ISO 9126 software quality metrics (Functionality) support by UML suite**. **SIGSOFT Softw. Eng. Notes**, 30(2):1–5, 2005. ISSN 0163-5948. doi: http://doi.acm.org/10.1145/1050849.1050860.

Eric Evans. **Domain Driven Design**. Addison-Wesley, 2004.

Erich Gamma, Richard Helm, Ralph Johnson, en John Vlissides. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley, 1999.

Jacob Kleerekoper. **Design of a Pattern Definition Language**. Master's thesis, Utrecht University, november 2007.

Kung-Kiu Lau en Zheng Wang. **Software Component Models**. **IEEE Transactions on Software Engineering**, 33(10):709–724, 2007. ISSN 0098-5589. doi: http://doi.ieeecomputersociety.org/10.1109/TSE.2007.70726.

John D. McGregor en Timothy D. Korson. **Integrated object-oriented testing and development processes**. **Commun. ACM**, 37(9):59–77, 1994. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/182987.184075.

Roger S. Pressman. **Software Engineering: A Practitioner's Approach**. McGraw-Hill, 2005.

Jeroen Snijders. **Functional Design Patterns**. Master's thesis, Utrecht University, augustus 2004.

van Dale. **Van Dale Internetwoordenboek**, 2008. URL `http://www.vandale.nl/`.

Mark R. Vigder en John Dean. **An architectural approach to building systems from COTS software components**. In **CASCON '97: Proceedings of the 1997 conference of the Centre for Advanced Studies on Collaborative research**, page 22. IBM Press, 1997.

Andere gebruikte bronnen

Robbert Beukema, Patrick van Driel, Robert Guitink, en Robbert Brak. **Handleiding Functionele documentatie voor de Quinity Polisadministratie v1.2**. Intern document, april 2007.

Yoran Maxim Bosman. **Incorporating functional design patterns in software development**. Master's thesis, Universiteit Twente, Augustus 2007.

Robert Guitink. **Omschrijving van het gebruik van Functional Design Patterns v1.0.1**. Intern document, Mei 2006.

Robert Guitink en Dennis Bron. **Functioneel Ontwerp - SDE Standards and Guidelines v1.5**. Intern document, Juli 2007.

Patrick van Driel. **Overzicht projectuitvoering v1.1**. Intern document, februari 2007.