Afstudeerscriptie Informatiekunde:

Use of viewpoints in system development

Naam:	Mehmet Anuk
Studentnummer:	s506257
Faculteit:	Faculteit der Natuurwetenschappen,Wiskunde en Informatica
	Instituut: Nijmeegs Instituut voor Informatica en Informatiekunde
Datum:	25 januari 2008
Versie:	Versie 5.8

Samenvatting

De voornaamste reden waarom veel ICT-projecten door klanten niet als succesvol wordt ervaren komt doordat opgeleverde systemen vaak niet aan klantverwachtingen voldoen. Incomplete requirements engineering en/of slechte communicatie tussen betrokkenen ligt vaak ten grondslag voor het niet voldoen aan klantverwachtingen. Als men ervoor wil zorgen dat systemen beter aan de klantverwachtingen voldoen, is het daarom belangrijk om de requirements engineering en de communicatie tussen betrokkenen te verbeteren.

Viewpoints zijn ervoor bedoeld om stakeholder perspectieven beter te kunnen structuren, organiseren en te managen. Het werken met viewpoints kan gezien worden als een middel om met het meerdere perspectieven probleem om te kunnen gaan. Wanneer men spreekt over het modelleren van viewpoints bedoelt men eigenlijk het modelleren van views. Een view is namelijk het resultaat van een viewpoint.

Momenteel zijn er verschillende beschrijvingen van het begrip viewpoint in de omloop zijn. Zo hanteren diverse wetenschappers en wetenschappelijke organisaties (in meer of mindere mate) een eigen methodiek en terminologie ten aanzien van het werken met viewpoints. Als gevolg hiervan zijn er door de loop der jaren tientallen verschillende view modellen en viewpoints ontwikkeld. Het grote aanbod aan view modellen en het verschil in werkmethodiek en terminologie maakt het kiezen van een view model niet eenvoudig. Classificatieraamwerken (zoals die zijn gepresenteerd door het Telematica Instituut in Enschede) kunnen de IT-architect of ontwikkelaar ondersteunen bij het maken van een viewpoint keuze.

Hoewel de view modellen onderling sterk van elkaar kunnen verschillen, is het achterliggende principe min of meer hetzelfde. De meeste modellen werken namelijk vanuit het belang van de stakeholder.

De centrale onderzoeksvraag is:
“ In hoeverre leidt het ontwikkelen van systemen aan de hand van viewpoints tot systemen die beter aan klantverwachtingen voldoen? ”

Er zijn genoeg aanwijzingen te vinden dat het werken met viewpoints een positieve bijdrage zal hebben op de systeemontwikkeling. Het ontbreken van voldoende praktijkervaring maakt het momenteel echter onmogelijk om deze bewering onweerlegbaar te bewijzen.

Voorwoord

Deze afstudeerscriptie is geschreven in het kader van de afronding van mijn studie informatiekunde aan de faculteit der Natuurwetenschappen, Wiskunde en Informatica (NWI) van de Radboud universiteit (RU) in Nijmegen.

Ten eerste wil ik graag mijn afstudeerbegeleider Dr. Patrick van Bommel hartelijk bedanken voor zijn actieve rol in het geven van feedback in de vorm van suggesties en het stellen van kritische vragen. Zijn begeleiding en het advies wat hij mij de afgelopen periode heeft gegeven heb ik als zeer waardevol en leerzaam ervaren. Verder wil graag Geeske Boerma en Zelal Anuk bedanken voor hun hulp bij het verrichten van de spelling en grammaticale controles op dit document.

Tenslotte gaat mijn dank uit naar iedereen die mij heeft geholpen en gesteund tijdens het afstudeertraject.

Mehmet Anuk
Arnhem, 25 januari 2008

Inhoudsopgave

1. Inleiding

1.1 Aanleiding

Regelmatig leest men in kranten of ziet men op het nieuws, dat er weer eens voor enorme bedragen een ICT-project is mislukt. Het falen van ICT-projecten is helaas geen zeldzaam fenomeen. Een aantal jaar geleden heb ik dit tijdens een systeemontwikkelingsproject zelf ook een keer meegemaakt. Door een gebrekkige communicatie tussen mij en de toekomstige gebruikers van het systeem, sloeg ik de plank volledig mis en leverde ik (tot mijn grote verbazing) een product op dat totaal niet aan hun verwachtingen voldeed. Als gevolg hiervan, kon ik het systeem in de prullenbak weggooien en moest ik als het ware weer helemaal opnieuw beginnen met het project. Om zulke rampzalige scenario's in de toekomst te vermijden, had ik mezelf voorgenomen om op zoek te gaan naar mogelijke oplossingen voor deze probleemstelling.

Tijdens de cursus Digitale Architectuur kwam ik voor het eerst in aanraking met het onderwerp viewpoints. Het gebruik van viewpoints leek me een zeer interessante mogelijkheid te bieden in mijn zoektocht naar een methodiek om producten op te kunnen leveren, die beter aan de klantverwachtingen voldoen. Ik heb destijds dan ook naar aanvullende informatie gezocht omtrent het werken met viewpoints. Het resultaat van mijn speurwerk was nogal teleurstellend en ik werd hier helaas niet veel wijzer van. Dit kwam doordat er relatief weinig literatuur over dit onderwerp beschikbaar was. Daarnaast was de informatie die ik kon vinden, nogal divers van karakter. Hierdoor had het eerder een verwarrende dan een opheldere werking op me. Toch bleef dit onderwerp gedurende mijn studie, mijn nieuwsgierigheid trekken. De afstudeerscriptie vormde voor mij de ideale kans om dit onderwerp beter te kunnen onderzoeken. Ik heb dankbaar gebruik gemaakt van de gelegenheid en heb deze kans dan ook met beide handen gegrepen.

1.2 Leeswijzer

Hieronder wordt in het kort toegelicht hoe deze afstudeerscriptie qua structuur is opgebouwd. Dit document bestaat uit drie delen, namelijk:

Deel I – Introductie

In het introductie gedeelte wordt de probleemstelling gedefinieerd, de verantwoording voor dit onderzoek gegeven, het theoretische kader voorgelegd en wordt er beschreven hoe het onderzoek is verricht. Het doel van dit deel is de lezer meer inzicht te geven in de onderzoekmethodiek.

- Hoofdstuk 2 - Probleemstelling
  In dit hoofdstuk wordt de centrale onderzoeksvraag gepresenteerd, komen de belangrijkste deelvragen aan bod en wordt de probleemstelling en de onderzoeksdoelstelling van deze afstudeerscriptie beschreven.

- Hoofdstuk 3 - Verantwoording
  In dit hoofdstuk geef ik de reden voor dit onderzoek aan. Zo worden onder andere de nieuwswaarde en theoretische, praktische en maatschappelijke relevantie van het onderzoek toegelicht.

- Hoofdstuk 4 – Theoretisch kader
  Hier wordt het kennisgebied van het onderzoek gespecificeerd, komen de relevante inhoudelijke keuzes en veronderstellingen aan bod en worden de belangrijkste termen en concepten die in dit onderzoek gebruikt worden toegelicht.

- Hoofdstuk 5 – Onderzoeksstructuur
  In dit hoofdstuk wordt de structuur van het onderzoek nader toegelicht. Zo zal de onderzoeksfunctie worden beschreven, worden de belangrijkste deelvragen behandeld en wordt er verteld hoe de centrale onderzoeksvraag beantwoord wordt.

Deel II – Data verzameling
In dit gedeelte wordt de benodigde kennis gegenereerd om de centrale onderzoeksvraag te kunnen beantwoorden. Het doel van dit deel is om de lezer meer inzicht te geven over het onderzoeksonderwerp.

- Hoofdstuk 6 – Waarom falen veel ICT-projecten?
  Dit hoofdstuk presenteert de hoofdproblematiek van dit onderzoek, namelijk waarom veel ICT-projecten falen en hoe het komt dat de opgeleverde resultaten niet aan klantverwachtingen voldoen.

- Hoofdstuk 7 – Viewpoints
  Hier wordt het onderwerp viewpoints uitvoerig toegelicht. Zo zal er onder andere worden verteld wat een viewpoint is, wat men ermee kan doen, hoe men ze gebruikt en welke belemmeringen men kan verwachten wanneer er met viewpoints gewerkt wordt. Ook worden een aantal view modellen behandeld, zodat er meer inzicht verstrekt wordt over de achterliggende gedachte van het werken met viewpoints.

- Hoofdstuk 8 – In hoeverre zijn viewpoints vernieuwend
  In dit hoofdstuk wordt een vergelijking gemaakt tussen het principe achter traditionele systeemontwikkelingmethoden en viewpoint-gebaseerde systeemontwikkelingmethodes. Zo kan men te weten komen in hoeverre viewpoints vernieuwend zijn in de systeemontwikkeling.

Deel III – Data analyse
In dit gedeelte worden alle deelvragen tezamen met de centrale onderzoeksvraag aan de hand van de informatie dat uit Deel II komt beantwoordt. Doelstelling van dit deel is om de onderzoeksscriptie af te kunnen ronden.

- Hoofdstuk 11 – Conclusie
  In dit hoofdstuk worden de deelvragen en de centrale onderzoeksvraag beantwoord. Verder worden er suggesties gedaan voor eventueel verder onderzoek en geef ik een persoonlijke evaluatie van mijn afstudeerperiode.

Deel I

Introductie

2. Probleemstelling

2.1 Onderzoeksvraag

De centrale onderzoeksvraag is als volgt gedefinieerd.

" In hoeverre leidt het ontwikkelen van systemen aan de hand van viewpoints tot systemen die beter aan klantverwachtingen voldoen? "

2.2 Deelvragen

De onderstaande deelvragen dienen beantwoord te worden om een antwoord op de centrale onderzoeksvraag te kunnen verkrijgen.

- Waarom falen veel ICT-projecten?
- Wat zijn viewpoints?
- Kunnen viewpoints gemodelleerd worden?
- In hoeverre is het werken met viewpoints betrouwbaar?
- In hoeverre leidt het gebruik van viewpoints tot een betere requirements engineering?
- In hoeverre verbeteren viewpoints de communicatie tussen betrokkenen?

2.3 Toelichting over de probleemstelling

Met het complexer worden van onze maatschappij, neemt niet alleen de vraag naar krachtigere systemen verder toe, maar worden ook de eisen en wensen die gebruikers aan deze systemen stellen uitgebreider. Voor ontwikkelaars is het vaak een enorme uitdaging om systemen op te leveren die aan alle klantverwachtingen voldoen.

Om het juiste product op te kunnen leveren, is goede communicatie van essentieel belang. Dit lijkt vanzelfsprekend, maar toch komt het in de praktijk [2, 9] regelmatig voor dat er tijdens systeemontwikkelingsprojecten niet duidelijk gecommuniceerd wordt tussen betrokkenen. Dit kan leiden tot situaties waarbij er systemen worden ontwikkeld die niet aan de klantverwachtingen voldoen en in het ergste geval zelfs tot projecten die volledig mislukken [3].

Alleen in Nederland gaan er jaarlijks al voor honderden miljoenen euro's [4, 5] aan mislukte ICT-projecten verloren. Vooral bij grote ICT-projecten die veel resources (in de vorm van tijd, geld en capaciteit) in beslag nemen, kunnen dit soort scenario's rampzalige gevolgen hebben voor de continuïteit van een organisatie.

2.4 Doelstelling van dit onderzoek

Het ontwikkelen van computersystemen wordt vaak in fases gedaan [6, 7]. Zo bestaat een traditioneel systeemontwikkelingtraject meestal uit een definitie-, ontwerp-, realisatie- en een implementatiefase. Ieder systeemontwikkelingtraject kent (ongeacht de methodiek die gehanteerd wordt) altijd wel een moment waarbij de eisen en wensen voor het nog te ontwikkelen systeem in kaart gebracht dienen te worden.

Het achterhalen en documenteren van systeemeisen wordt binnen de systeemontwikkeling ook wel requirements engineering [52] genoemd. Middels dit onderzoek wil ik analyseren of het gebruik van viewpoints bijdraagt tot een betere requirements engineering en zo ook leidt tot de ontwikkeling van producten (systemen) die beter aan de klantverwachtingen voldoen. Uiteindelijk probeer ik met mijn onderzoek een bijdrage te leveren aan de verbetering en optimalisering van het systeemontwikkelingsproces in het algemeen.

2.5 Afbakening van de onderzoeksdoelstelling

De scope van dit onderzoek beperkt zich uitsluitend tot het beantwoorden van de centrale onderzoeksvraag. Dit betekent dat er met uitzondering van viewpoint-gebaseerde systeemontwikkelingmethoden, verder geen andere alternatieven zullen worden onderzocht die het systeemontwikkelingsproces en in het bijzonder de requirements engineering zouden kunnen verbeteren.

3. Verantwoording

3.1 Waarom dit onderzoek?

Ieder ICT-project loopt afhankelijk van zijn omvang en complexiteit, het risico een resultaat op te leveren dat niet aan de klantverwachtingen voldoet. In sommige gevallen kan zelfs het project hierdoor mislukken. Bij grote ICT-projecten waar de financiële kosten al gauw in de miljoenen euro's kunnen lopen, zijn dergelijke rampscenario's absoluut onacceptabel.

Voor het slagen van een ICT-project is goede communicatie tussen ontwikkelaars en overige stakeholders van essentieel belang. Bijna iedere projectmanager is zich hier van bewust en toch vormt gebrekkige communicatie tussen betrokkenen, één van belangrijkste factoren voor het mislukken van ICT-projecten. Ongeveer 20% van alle mislukte ICT-projecten in Nederland valt hieraan te verwijten [8] [9].

Om het risico op onbruikbare producten die niet aan de klantverwachtingen voldoen te verkleinen, is het zeer belangrijk om de systeemeisen (requirements) duidelijk in kaart brengen. Het achterhalen van requirements kan echter een zeer kostbaar en tijdrovend proces zijn [10].

Viewpoints kunnen de ontwikkelaar ondersteuning bieden bij het herkennen van systeemeisen en vereenvoudigen daarnaast het werken met verschillende perspectieven van stakeholders. Hierdoor zouden er mogelijk veel kostbare resources bespaard worden.

Gezien de vaak hoge bedragen die er met ICT-projecten gemoeid zijn, zou het een gemiste kans zijn om dit niet te onderzoeken.

3.2 Nieuwswaarde

In de afgelopen jaren is het gebruik van computersystemen zowel in het bedrijfsleven als in de (semi) publieke sector sterk toegenomen. De verwachting is dat de vraag naar computersystemen en zo ook het aantal ICT-projecten de aankomende jaren verder zal blijven groeien [11, 12]. Deze ontwikkeling zal ongetwijfeld een positieve invloed op de nieuwswaarde van dit onderzoek hebben.

3.3 Theoretische relevantie

Het begrip viewpoint is nog relatief onbekend in de systeemontwikkeling. Zo zijn er tot op heden (in vergelijking tot andere vakgebieden binnen de informatiekunde), weinig onderzoeken over dit onderwerp verricht en is de terminologie op dit gebied nog verre van gestandaardiseerd.

Het gebrek aan concrete kennis over viewpoint-gebaseerde systeemontwikkeling-methoden, komt deze methodiek dan ook niet echt ten goede. Hoe minder kennis en ervaring er aanwezig is, des te minder de waarschijnlijkheid zal zijn dat viewpoint-gebaseerde systeemontwikkelingmethoden door systeemontwikkelaars en IT-architecten in de praktijk zullen worden toegepast.

Daarom is het belangrijk dat er meer onderzoek over dit onderwerp wordt verricht, zodat de benodigde kennis, ervaring en vertrouwen ontstaat om deze theorieën ook in de praktijk te doen realiseren. Daarnaast kan dit onderzoek tevens ook een basis bieden voor de ontwikkeling van nieuwe theorieën op het gebied van viewpoint-gebaseerde systeemontwikkeling en kunnen bestaande viewpoint-theorieën eventueel geëvalueerd en verbeterd worden.

3.4 Praktische relevantie

Indien uit dit onderzoek blijkt dat er voldoende aanwijzingen zijn dat het werken met viewpoints een gunstig effect zal hebben op de het systeemontwikkelingsproces in het algemeen, dan kan de kennis die uit dit onderzoek wordt gewonnen door anderen gebruikt worden om viewpoint-gebaseerde systeemontwikkelingmethoden in de praktijk toe te passen.

3.5 Maatschappelijke relevantie

In ons dagelijks leven worden we (bewust of onbewust) steeds afhankelijker van computersystemen. Ontwikkelingen die een positieve ofwel negatieve invloed op de systeemontwikkeling kunnen hebben, dienen daarom te worden onderzocht.

3.6 Doelgroep

De beantwoording van de centrale onderzoeksvraag zal in het bijzonder voor systeemontwikkelaars en projectmanagers interessant zijn. Aan hand van het resultaat uit dit onderzoek kunnen zij bestaande systeemontwikkelingmethodieken evalueren en indien nodig verbeteren.

4. Achtergrond: theoretisch kader

4.1 Specificatie kennisgebied

Het kennisgebied spreidt zich over een tweetal verschillende wetenschappelijke vakgebieden uit, namelijk: de sociale wetenschappen [13] en de informatiekunde [14]. De reden dat het kennisgebied over (tenminste) twee disciplines verspreid is, komt doordat systeemontwikkeling op zichzelf een zeer breed en complex onderwerp is.

Waarom sociale wetenschappen?

De menselijke factor is zonder enige twijfel zeer belangrijk binnen de systeemontwikkeling. Zo is men tijdens de requirements engineering in grote mate afhankelijk van de input die door mensen gegeven wordt, kunnen de stakeholder belangen door sociale omstandigheden beïnvloed worden, wordt er binnen systeemontwikkelings-projecten vaak in teamverband samengewerkt en kunnen communicatieve en sociale factoren het systeemontwikkelingtraject sterk beïnvloeden.

Systeemontwikkeling is dus meer dan alleen ontwerpen en programmeren. Naast kennis en informatie, spelen ook communicatieve, psychologische en sociale aspecten een zeer belangrijke en vooral niet te onderschatten rol.

Waarom informatiekunde?

Onderwerpen op het gebied van systeemontwikkeling en requirements engineering behoren van oudsher tot de informatiekunde. Vraagstellingen hierover komen regelmatig voor in de informatiekunde en zijn daardoor zeer typerend voor dit vakgebied.

Om een antwoord op de centrale onderzoeksvraag te kunnen verkrijgen, dient dit onderzoek dan ook nadrukkelijk vanuit een informatiekundig perspectief bekeken te worden. De informatiekunde vormt namelijk een onmisbare schakel tussen mens en systeem.

Inperking kennisgebied

Hoewel er elementen uit de sociale wetenschappen aanwezig zijn, heeft dit onderzoek zijn raakvlakken voornamelijk in de informatiekunde. Wanneer het kennisgebied vanaf dit punt verder wordt ingeperkt, kan men concluderen dat de huidige probleemstelling vooral betrekking heeft op het onderzoeksthema “requirements engineering”. Samengevat is de stapsgewijze inperking van het kennisgebied als volgt te definiëren:

Informatiekunde → Systeemontwikkeling → Requirements Engineering (→ Viewpoints).

4.2 Relevante inhoudelijke keuzes en veronderstellingen

Gedurende dit onderzoekstraject zullen de verschillende viewpoint-gebaseerde systeemontwikkelingmethoden inhoudelijk gezien niet geanalyseerd worden, omdat de benodigde tijd en kennis hiervoor simpelweg ontbreekt. Zo vallen opmerkingen en verbeterpunten over individuele viewpoint-gebaseerde systeemontwikkelingmethoden buiten het bereik van dit onderzoek.

De primaire doelstelling van dit onderzoek is namelijk niet om de huidige viewpoint-gebaseerde systeemontwikkelingmethodes te verbeteren, maar om meer inzicht verstrekken in het principe achter viewpoint-gebaseerde systeemontwikkeling.

4.3 Belangrijkste termen en concepten

Hieronder bevindt zicht een overzicht van de belangrijkste termen en concepten die in dit onderzoek gehanteerd worden. Een groot gedeelte van de onderstaande gebruikte terminologie komt uit ISO 1471 [15]. Voor meer informatie over ISO 1471 wordt verwezen naar hoofdstuk zeven van deze afstudeerscriptie.

Domein: Systeemontwikkeling (in het bijzonder de requirements engineering)

Variabelen:
- Systeem
- Missie
- Stakeholder
- View
- Model
- Library Viewpoint
- View model
- Environment
- Architectuur
- Concern
- Viewpoint
- Rationale
- Architectuurbeschrijving

Beschrijving van de variabelen:

Variabelen	Beschrijving	Voorbeeld waarden	Meetniveau
Systeem	Het systeem waarvan de architectuur beschreven wordt.	TISS, Blackboard, Microsoft Windows, SAP, etc.	nominaal
Environment	Omgeving waarin het systeem functioneert.	Afdeling, bedrijf, onderwijsinstelling, etc.	nominaal
Missie	Doel van het systeem binnen het environment.	Registeren van storingen, maken van back-ups, verzorgen van email, etc.	nominaal
Architectuur	De architectuur van het systeem.	Enterprise architectuur	nominaal
Architectuur-beschrijving	Verzameling documenten die samen de architectuur beschrijven.	Handleidingen, beleidsdocumenten, technische documenten,etc.	nominaal
Stakeholder	Persoon of organisatie (actor) die belang heeft bij het systeem.	Opdrachtgever, systeemontwikkelaar, gebruiker, etc.	nominaal
Concern	Belang (of zorg) van stakeholder.	Hoe blijf ik geïnformeerd?, Hoe maak ik de juiste keuzen? Hoe communiceer ik	nominaal

		effectief?, Welke diensten worden er verlangd van de andere rollen/functies.	
View	Onderdeel van de architectuurbeschrijving.	DFD model, ORM model, UML model, etc.	nominaal
Viewpoint	Voorschrift voor een view (inhoud en te gebruiken modellen, relatie naar concerns en stakeholders).	Technology viewpoint, Enterprise viewpoint, Business Function viewpoint, etc.	nominaal
Model	Stuk gestructureerde informatie dat wordt gerealiseerd aan de hand van een bepaald representatie met eventueel een instructie voor het opstellen van het model.	DFD model, ORM model, UML model, etc.	nominaal
Rationale	Uitleg over de architectuurbeschrijving (in het bijzonder welke viewpoints er gekozen zijn en waarom).	Een tekstuele verklaring.	nominaal
Library Viewpoint	Viewpoint wat is vastgelegd voor gebruik in meerdere architectuurbeschrijvingen.	Technology viewpoint, Organisatorische viewpoint, Business Function viewpoint, etc.	nominaal
View model	Een view model bestaat uit een verzameling van voorgeschreven viewpoints op een domein (architectuur of systeem).	RM-ODP, Zachman raamwerk, Kruchten 4+1, etc.	nominaal

4.4 Theorieën met betrekking tot de probleemstelling

Hieronder worden enkele achtergrondtheorieën besproken, die naar mijn mening de oorzaak van de huidige probleemstelling mogelijk kunnen verklaren. Deze theorieën zijn echter (nog) niet bewezen. Uit dit onderzoek moet blijken in hoeverre de onderstaande theorieën kloppen. Door deze theorieën alvast hieronder te presenteren, hoop ik de lezer meer inzicht te kunnen geven in mijn persoonlijke gedachtegoed met betrekking tot de huidige probleemstelling.

Mogelijke oorzakelijke theorieën

- De huidige requirement engineering technieken zijn niet toereikend genoeg om een compleet beeld te kunnen krijgen van de stakeholder requirements. Zo zijn ze

niet in staat om rekening te houden met stakeholder concerns. Als men niet zeker weet wat de klant wil, ontstaat er een grotere kans dat producten niet aan de klantverwachtingen voldoen.

Verwachtingen ten aanzien van het onderzoeksresultaat

- Het gebruik van viewpoints en dus ook het werken met meerdere perspectieven zal de systeemontwikkelaar of IT-architect ondersteunen het te onderzoeken domein beter te begrijpen.

- Het gebruik van veel verschillende viewpoints zal het systeemontwikkelingsproces complexer en moeilijker te beheersen maken. Hoe meer viewpoints er zijn des te meer inspanning er nodig is om alles in goede banen te kunnen geleiden.

5. Onderzoeksstructuur

5.1 Onderzoeksfunctie

Een onderzoek bevat vaak meerdere onderzoeksfuncties, toch zal er altijd één onderzoeksfunctie dominant zijn. Dit onderzoek is opgesteld vanuit een de verklarende functie.

5.2 Belangrijkste deelvragen

De onderstaande deelvragen dienen beantwoord te worden om een antwoord op de centrale onderzoeksvraag te kunnen verkrijgen.

Deelvragen

- Waarom falen veel ICT-projecten?
- Wat zijn viewpoints?
- Kunnen viewpoints gemodelleerd worden?
- In hoeverre is het werken met viewpoints betrouwbaar?
- In hoeverre leidt het gebruik van viewpoints tot een betere requirements engineering?
- In hoeverre verbeteren viewpoints de communicatie tussen betrokkenen?

5.3 De deelvragen nader toegelicht

Hieronder wordt per deelvraag de onderzoeksfunctie en doelstelling van de vraag binnen het onderzoek beschreven.

Deelvraag	Onderzoeksfunctie	Doelstelling
Waarom falen veel ICT-projecten?	Verklarend	Hiermee wordt achtergrond informatie verstrekt over de belangrijkste oorzaken voor het mislukken van ICT-projecten en het opleveren van producten die niet aan klantverwachtingen voldoen. Het onderzoek wordt hiermee gestart.
Wat zijn viewpoints?	Beschrijvend	Dit vormt één van de belangrijkste deelvragen uit het onderzoek. Aan de hand van deze vraag leert men het begrip viewpoint beter kennen en ziet men wat de mogelijkheden ermee zijn.
Kunnen viewpoints gemodelleerd worden?	Verklarend	Om te kunnen achterhalen of men aan de klantverwachtingen voldoet, dient men met stakeholders te communiceren. Het daarom belangrijk om te weten of viewpoints

		gemodelleerd kunnen worden. Modellen vormen namelijk een zeer geschikt middel om met mensen te communiceren.
In hoeverre is het werken met viewpoints betrouwbaar?	Verklarend	Om de vergelijking tussen traditionele systeemontwikkeling-methoden te kunnen maken, is het belangrijk om te weten in hoeverre viewpoint-gebaseerde methoden betrouwbaar zijn.
In hoeverre leidt het gebruik van viewpoints tot een betere requirements engineering?	Verklarend	Een goede requirements engineering is een zeer belangrijke randvoorwaarde om producten op te kunnen leveren, die aan klantverwachtingen voldoen. Als het werken met viewpoints de requirements engineering verbetert, zal dit ook waarschijnlijk een positieve invloed hebben op de kans dat een product aan de klantverwachtingen voldoet.
In hoeverre verbeteren viewpoints de communicatie tussen betrokkenen?	Verklarend	Een goede communicatie tussen betrokkenen is een zeer belangrijke randvoorwaarde om producten op te kunnen leveren die aan klantverwachtingen voldoen. Als het werken met viewpoints de communicatie verbeterd, zal dit ook waarschijnlijk een positieve invloed hebben op de kans dat een product aan de klantverwachtingen voldoet.

5.4 Beantwoording onderzoeksvraag

De centrale onderzoeksvraag is in deelvragen opgesplitst. Dit maakt het oplossen van de centrale onderzoeksvraag een stuk eenvoudiger. Iedere deelvraag legt namelijk een stukje van het antwoord op de centrale onderzoeksvraag bloot.

Het moge duidelijk zijn dat deze deelvragen eigenlijk op zichzelf staande subonderzoeken zijn. Wanneer de resultaten uit deze subonderzoeken bij elkaar worden gevoegd, ontstaat er voldoende materiaal, kennis en inzicht om de centrale onderzoeksvraag te kunnen beantwoorden.

Deel II

Data verzameling

6. Waarom falen veel ICT-projecten?

Anno 2007 is het nog steeds een tragisch feit dat het overgrote deel van de ICT-projecten niet als succesvol worden ervaren of zelfs mislukt [16, 17, 25]. Zo worden projecten vaak te laat afgerond, blijken ze financieel meer te kosten dan er oorspronkelijk begroot is, worden ze vroegtijdig gestopt of voldoet het resultaat niet aan de verwachtingen.

6.1 Wanneer is een project succesvol?

Het is niet eenvoudig om succes te meten. Een welbekend manier om de mate van succes enigszins te bepalen, is het hanteren van succescriteria [18, 19, 20, 26]. Er bestaan een groot aantal succescriteria, maar om het overzichtelijk te houden zullen hieronder alleen de belangrijkste drie criteria gedefinieerd worden.

Een ICT-project dient op z'n minst aan de volgende criteria [21, 17] te voldoen om als succesvol gekenmerkt te kunnen worden.

- Het systeem is op tijd geleverd;
- Het systeem is volgens het begrootte bedrag (of voor minder) gerealiseerd;
- Het systeem werkt zoals verwacht of vereist.

In de praktijk [22] blijken projectmanagers echter vaak moeite te hebben om aan alle drie de criteria tegelijkertijd te voldoen. Het komt dan ook regelmatig voor, dat er uiteindelijk bij de beëindiging van een ICT-project één of meerdere criteria niet worden behaald.

6.2 De belangrijkste factoren voor succes of mislukking

Er zijn in het verleden verschillende onderzoeken [16, 17, 23, 24] gedaan naar zowel geslaagde als mislukte ICT-projecten. Uit deze onderzoeken is gebleken dat er geen alles overtreffende factor bestaat voor het doen slagen of falen van een ICT-project. Zo kunnen meerdere factoren verantwoordelijk zijn voor het mislukken van een project en is het zelfs mogelijk dat factoren elkaar beïnvloeden.

Het aantal factoren, dat van invloed kan zijn op het resultaat van een ICT-project is als het ware onbeperkt. Dit maakt het voor projectmanagers soms uitermate moeilijk om projecten succesvol te kunnen managen. Moeilijk betekent echter niet onmogelijk.

Wanneer men de oorzakelijke aspecten van meerdere mislukte ICT-projecten met elkaar vergelijkt, komt men al gauw tot de conclusie dat sommige factoren herhaaldelijk voorkomen. Op deze manier is het mogelijk om de meest voorkomende aspecten, die van invloed kunnen zijn op het falen van een ICT-project te achterhalen. Zodra men deze factoren eenmaal in kaart heeft gebracht, wordt het voor de projectmanager een stuk eenvoudiger om hier tijdens een ICT-project mee rekening te houden en de kans op een succesvol eindresultaat te vergroten.

Op de volgende pagina bevindt zich een opsomming van de meest voorkomende en zo ook de belangrijkste factoren die van invloed kunnen zijn op het succes van een ICT-project. Deze factoren zijn op basis van willekeurigheid gerangschikt en zijn voor een deel

gebaseerd op de mate van frequentie waarop ze in diverse onderzoeksresultaten [16, 17, 23, 24] als oorzakelijk aspect gekenmerkt worden.

- Gebrekkige betrokkenheid van gebruikers
  Onvoldoende betrokkenheid van gebruikers kan fataal zijn voor het eindresultaat van een ICT-project. Zo is het zeer onwaarschijnlijk dat toekomstige gebruikers, enige loyaliteit (of commitment) naar een systeem zullen tonen, wanneer ze niet direct bij de het project zelf betrokken worden. In het ergste geval kan dit zelfs tot vijandig gedrag tegenover het nieuwe systeem leiden.

  Hierdoor is het belangrijk om gedurende het project zowel het management als de toekomstige gebruikers bij de ontwikkeling van het systeem te betrekken. Op deze manier verkleint men de weerstand en vergroot men de kans op een product dat beter aan de verwachtingen van toekomstige gebruikers voldoet.

- Lange of onrealistische tijdsplanningen
  Organisaties zijn dynamisch van karakter. Dit heeft als gevolg dat gedurende een ICT-project gebruikerseisen en -wensen kunnen veranderen. Wanneer een project te lang duurt, loopt men hierdoor het risico dat tegen de tijd dat het systeem uiteindelijk klaar is, deze niet meer aan de gebruikersbehoeften voldoet.

  Daarom dienen planningen kort te zijn en is het aan te raden om bij de ontwikkeling van grote computersystemen, het project in kleinere subprojecten te splitsen. Veel projectmanagers zijn zich bewust van dit risico en proberen hun planningen dan ook zo kort mogelijk te houden.

  Men doet er verstandig aan om de situatie eerst goed te analyseren. Zo is het belangrijk om ervoor uit te kijken dat er geen onrealistische tijdsplanningen worden opgesteld. Wanneer planningen niet realistisch zijn, zullen projecten vertragingen oplopen of moeten er uiteindelijk systeemonderdelen geannuleerd worden om alsnog de oorspronkelijke planning te kunnen redden. Deze ad-hoc “last minute” handelingen kunnen ten koste gaan van de gebruikers verwachtingen en de kwaliteit van het eindproduct. Daarom is het zeer belangrijk om projectplannen van te voren goed op hun haalbaarheid te analyseren.

- Slechte of geen requirements
  ICT-projecten hebben vaak met vage requirements te maken. Zo is een veel voorkomend probleem, dat toekomstige gebruikers hun eisen en wensen met betrekking tot de functionaliteiten van het te ontwikkelen systeem niet concreet naar de ontwikkelaars kunnen overbrengen. In het ergste geval is het zelfs mogelijk dat de gebruiker helemaal niet weet wat hij of zij wil.

  Wanneer systeemontwikkelaars geen duidelijke input van de toekomstige gebruikers ontvangen, kan dit leiden tot scenario’s waarbij de ontwikkelaars zelf (zonder dat ze enige diepgaand kennis over het domein hebben) gaan bouwen wat ze denken nodig te hebben. Hierdoor lopen ze het risico een product op te leveren dat uiteindelijk niet aan de klantverwachtingen voldoet.

Voor het slagen van ICT-projecten is het verrichten van een complete requirements engineering daarom van essentieel belang. Als een gebruiker niet precies weet wat hij of zij wil, is het namelijk de taak van de requirements engineer om deze onduidelijkheid over te zetten naar duidelijkheid.

- Zeer flexibele productscope
  Het begrip scope wordt gebruikt om aan te duiden welke functionaliteiten het ontwikkelde systeem uiteindelijk dient te hebben. Wanneer er sprake is van een zeer flexibele productscope, houdt dit in dat gedurende het project, de functionele eisen van het systeem zeer eenvoudig kunnen veranderen of toenemen. Dit betekent dat gebruikers tijdens de ontwikkeling van het systeem telkens met nieuwe functionele eisen en wensen kunnen komen. Hoe meer functies er gemaakt dienen te worden, des te meer kosten er gemaakt worden en het langer zal duren voordat het systeem af is.

  Als de functionele aspecten van het systeem niet worden afgebakend, loopt men het risico dat het project over de financiële begroting gaat en dat de oorspronkelijke tijdsplanning uiteindelijk zal wordt overschreden. Daarom dient de projectmanager ervoor te zorgen dat de eisen en wensen met betrekking tot de functionaliteiten van het systeem vooraf goed worden vastgelegd en dat iedereen zich eraan houdt totdat het systeem af is. Dit klinkt misschien eenvoudig, maar in de praktijk kan het tegenvallen om dit te realiseren. De gebruiker is namelijk in veel gevallen tevens ook de klant of opdrachtgever. Dit maakt het voor een ontwikkelaar soms extra moeilijk om “nee” te zeggen.

- Geen veranderingsmanagement systematiek
  Mede veroorzaakt door de dynamische maatschappij waarin we leven zijn organisaties continu aan veranderingen onderhevig. Hierdoor dient men gedurende een ICT-project er vanuit te gaan dat requirements ook vatbaar zijn voor verandering.

  Wanneer veranderingen op een ongecontroleerde wijze worden verricht, kan dit negatieve gevolgen voor het eindresultaat van een project hebben. Het is daarom belangrijk een vorm van veranderingsmanagement te hanteren. Op deze manier kan men (rekeninghoudend met de huidige begroting en planning) analyseren in hoeverre veranderingen mogelijk zijn binnen het project. Indien de beschikbare resources te kort schieten, dient men te achterhalen wat er nodig is om deze veranderingen toch te realiseren. Als er teveel resources in beslag worden genomen en het hierdoor niet mogelijk is om de gewenste veranderingen toe te passen, kan de veranderingsverzoek worden geweigerd. Wanneer veranderingsverzoeken wel reëel en uitvoerbaar zijn, kunnen deze vervolgens op een gecontroleerd wijze worden gerealiseerd.

  Welke veranderingsmanagementtechnieken men ook besluit te hanteren, het is en blijft belangrijk dat de projectmanager nooit het overzicht verliest. Een onvoorspelbaar ontwikkelingsproces komt namelijk niet ten goede aan de kansen op een succesvol eindresultaat.

- Niet voldoende getest
  Gedurende de ontwikkeling van een systeem zullen de ontwikkelaars ongetwijfeld een hoop testwerk moeten verrichtten om het systeem op fouten en prestaties te controleren. Vaak blijken deze tests niet voldoende te zijn en dient er door de toekomstige gebruikers ook een acceptatietest uitgevoerd te worden om te kunnen achterhalen of het systeem aan hun verwachtingen en eisen voldoet.

  Het komt regelmatig voor dat een acceptatietest niet alle fouten boven water legt voordat het systeem live gaat. Een aantal veel voorkomende oorzaken hiervoor zijn:

  - Er slechte requirements zijn, die niet geverifieerd en getest kunnen worden;
  - De testen slecht voorbereid zijn en methodisch niet orde zijn;
  - Te weinig tijd is geweest om adequaat te kunnen testen;
  - De testers onvoldoende getraind en ervaren zijn.

  Een ICT-project kan pas echt succesvol worden genoemd, wanneer het resultaat naar behoren werkt. Als het systeem na de implementatie nog steeds ernstige fouten bevat, zal dit ten koste gaan van de bruikbaarheid ervan. Deze fouten dienen (mits de opdrachtgever dat wil) verbeterd te worden. De handelingen die hierop volgen, zullen extra resources in beslag nemen, waardoor het project uiteindelijk toch de oorspronkelijke begroting en tijdsplanning zal overschrijden.

  Testen is en blijft een moeilijk proces. Vragen zoals “Hoeveel testen zijn genoeg?" en “hebben we al alle fouten ontdekt?” zijn niet eenvoudig te beantwoorden, dit verschilt namelijk per situatie. Vaak wanneer projecten dreigen uit te lopen, neigen mensen de verloren tijd in te halen door het testproces in te korten. Gezien de nadelige consequenties die een dergelijke keuze als deze met zich mee kan brengen, doet men er verstandig aan om het testproces niet te onderschatten.

- Gebrekkige communicatie
  Goede communicatie tussen betrokkenen is van essentieel belang voor het slagen of falen van een ICT-project. Communicatie is een overheersende factor dat op alle overige factoren van invloed kan zijn. Helaas komt het nog steeds vaak voor dat systeemontwikkelaars zowel onderling als naar gebruikers toe gebrekkig communiceren, waardoor ze elkaar niet altijd goed begrijpen.

  Gebrekkige communicatie tussen betrokkenen kan er toe leiden dat er verkeerde keuzes gemaakt worden en het project uiteindelijk hierdoor mislukt. Daarom is het belangrijk dat er duidelijke afspraken, richtlijnen en werkprocedures worden opgesteld. Op deze manier kan het communicatieproces binnen het project beter ondersteund worden.

Wanneer men de bovenstaande factoren analyseert en met elkaar vergelijkt, valt het op dat bepaalde factoren elkaar (direct of indirect) kunnen beïnvloeden. Zo kan het ontbreken van goed veranderingsmanagement tot een zeer flexibele productscope leiden en kan gebrekkige communicatie tussen betrokkenen in combinatie met het besluit om

toekomstige gebruikers niet direct bij het project te betrekken in een slechte requirements engineering resulteren.

Dit zijn maar een paar eenvoudige voorbeelden van hoe diverse factoren elkaar kunnen beïnvloeden binnen een ICT-project. Men moet zich voorstellen dat hoe groter en complexer een project is, des te moeilijk het zal zijn om deze factoren te kunnen managen. Elk van de beschreven factoren kan (indien er geen rekening met ze gehouden wordt) de drie succescriteria, die eerder in dit hoofdstuk zijn beschreven zonder enige moeite overschrijden. Een projectmanager doet er dan ook verstandig aan om gedurende het project met deze factoren rekening te houden. Men dient zich wel te realiseren dat hoewel het een stap in de goede richting is, dit echter geen succesvol eindresultaat van een ICT-project kan garanderen. Het aanbieden van een 100% succesgarantie dat voor iedere project geldt is naar mijn mening namelijk onmogelijk.

6.3 Voornaamste oorzaak voor ontevredenheid

Volgens een recent onderzoek dat door Ernst & Young in 2007 [16] is verricht, blijkt dat ruim de helft van de ondervraagden IT-managers en directeuren in Nederland niet tevreden is over het eindresultaat van een onlangs gerealiseerde ICT-project. (Dit betekent echter niet dat ruim de helft van de ICT-projecten volledig is mislukt, maar dat deze als minder succesvol worden ervaren.)

De voornaamste reden voor deze ontevredenheid wordt veroorzaakt doordat men vindt dat het opgeleverde resultaat niet (volledig) aan de verwachtingen voldoet. Hieruit kan men concluderen dat het opleveren van producten die niet aan klantverwachtingen voldoen een veel voorkomend probleem is en dat het tevens als één van de belangrijke redenen kan worden gezien, waarom veel ICT-projecten mislukken of als minder succesvol worden ervaren.

Hoe komt het eigenlijk dat opleverde systemen in veel gevallen niet aan (klant of gebruikers) verwachtingen voldoen?

Incomplete requirements engineering

De meest voor de handliggende reden hiervoor is dat er een slechte requirements engineering is verricht. Deze bewering wordt niet alleen door de resultaten uit het onderzoek van Ernst & Young bevestigd, maar ook door die van de Standish Group [17, 25]. Zo is uit dit onderzoek gebleken, dat “33% van alle software projecten in Verenigde Staten door een incomplete requirements engineering mislukt”.

Op basis van de requirements wordt namelijk een systeem ontwikkeld en als de requirements niet correct zijn gedefinieerd, is het vanuit een praktisch oogpunt gezien bijna onmogelijk om alsnog (volledig) aan de klantverwachtingen te voldoen. Men dient er rekening mee te houden dat voor het verrichten van een goede requirement engineering, ook de toekomstige gebruikers tot op een zekere hoogte direct bij het project zelf betrokken dienen te worden.

Slechte requirements verhogen dus het risico op producten die niet aan klantverwachtingen voldoen. Om aan de klantverwachtingen te kunnen voldoen is het

belangrijk om te weten wat de klant wil. Gezien het feit dat veel ICT-projecten door incomplete requirements engineering mislukken, laat zien dat het verrichtten van een goede requirements engineering een behoorlijke uitdaging kan zijn. Dit wordt mede veroorzaakt door de volgende factoren [36]:

- In veel gevallen is de requirements engineer zelf geen domein expert. Veel problemen die gerelateerd zijn aan het formuleren van requirements wordt veroorzaakt door misvattingen tussen requirements engineers, software engineers en impliciete assumpties door toekomstige gebruikers.

- Vanwege het verschil aan evaring en educatie zijn toekomstige gebruikers en requirements engineers niet altijd op de hoogte van de terminologie die door beiden partijen gebruikt worden, dit kan de communicatie tussen beiden partijen aanzienlijk belemmeren.

- Het begrip “compleetheid” is binnen de requirements engineering behoorlijk problematisch. Er bestaat namelijk geen eenvoudige analytische procedure voor het bepalen of gebruikers de requirements engineer alles hebben verteld wat ze nodig hebben om het systeem te kunnen maken.

- Requirements zijn nooit stabiel. Veranderingen in de omgeving waarin het systeem zal werken, kunnen de requirements van het systeem doen aanpassen. Het kan zelfs gebeuren dat veranderingen in een omgeving ervoor kunnen zorgen dat de requirements veranderen nog voordat het systeem überhaupt geïnstalleerd is.

- Vaak wordt er natuurlijke taal gebruikt om requirements te beschrijven. Hoewel deze manier van communicatie toekomstige gebruikers zal helpen het systeem beter te begrijpen, is er het probleem dat ambiguïteit (mede veroorzaakt door de rijkheid van een natuurlijke taal) naar misinterpretaties kan leiden.

- Er is geen eenzijdige aanpak of techniek die alle requirements van een systeem in één keer adequaat kan definiëren. Er is vaak meer dan één specificatietaal of -techniek nodig om requirements adequaat te kunnen definiëren.

Het ontwikkelen van producten die aan de klantverwachtingen voldoen is helaas niet zo vanzelfsprekend als men zou denken. De uitvoering van een goede en vooral complete requirements engineering vormt een essentieel randvoorwaarde om systemen op te kunnen leveren die aan de klant verwachtingen voldoen. Projectmanagers en requirements engineers doen er daarom verstandig aan om gedurende een ICT-project met de bovenstaande factoren rekening te houden.

Wanneer men de factoren analyseert die de requirements engineering zouden kunnen belemmeren, komt men al gauw tot de conclusie dat een aantal van deze factoren vooral communicatie gerelateerd zijn. Zo kom ik aan op de tweede factor die een rol van enige betekenis kan spelen waardoor veel opgeleverde systemen uiteindelijk niet aan de (klant) verwachtingen voldoen.

Gebrekkige communicatie tussen betrokken

Zo'n 20% [9] van alle mislukte ICT-projecten in Nederland valt aan communicatieve aspecten te verwijten en in de Verenigd Koninkrijk was dit percentage in 1998 zelfs 57% [27]. Wanneer er niet duidelijk gecommuniceerd wordt binnen een project, loopt men een groter risico dat betrokkenen elkaar verkeerd begrijpen en er hierdoor verkeerde keuzes gemaakt worden, waardoor de kans uiteindelijk groter wordt dat er producten worden ontwikkeld die niet aan de klantverwachtingen voldoen.

Communicatie is een heel breed aspect. Het toelichten van alle communicatie gerelateerde problemen die zich tijdens een ICT-project kunnen voordoen zal tot vele honderden mogelijke scenario's kunnen leiden. Aangezien dit geen noemenswaardige meerwaarde voor dit onderzoek levert, zullen deze scenario's dan ook niet worden toegelicht. Toch is het belangrijk om het één en ander te weten met betrekking tot de invloed, dat communicatie kan hebben ten aanzien van het opleveren van producten die aan klantverwachtingen voldoen.

Hoe meer mensen er bij de ontwikkeling van een systeem betrokken zijn, des te moeilijker het wordt om goed te communiceren. Zo kunnen er binnen een projectgroep meerdere mensen uit verschillende expertises werkzaam zijn. Afhankelijk van de taak en achtergrond van een bepaalde persoon, heeft iedere projectmedewerker zijn eigen (unieke) werkwijze, visie of belang ten aanzien van het te ontwikkelen systeem. Echter om het product af te kunnen leveren, dienen alle leden van de projectgroep met elkaar samen te werken. Omdat visies en belangen per persoon kunnen verschillen stijgt hierdoor ook de kans op misvattingen tussen projectleden onderling. Ik zal hier later in het volgende hoofdstuk op terugkomen.

Het kan ook gebeuren dat ontwikkelaars onderling wel goed met elkaar kunnen communiceren en juist naar toekomstige gebruikers niet. Dit kan er toe leiden dat ontwikkelaars gedurende een project denken dat ze het juiste product bouwen, terwijl ze in werkelijkheid een systeem ontwikkelen dat helemaal niet aan de gebruikers-verwachtingen voldoet. Vaak komen zulke communicatieve fouten pas tegen het einde van een project (bijvoorbeeld tijdens de acceptatietest) boven water. Wanneer ontwikkelaars regelmatig en op een duidelijke wijze naar toekomstige gebruikers communiceren, kunnen dergelijke scenario's wellicht voorkomen worden.

Voor het opleveren van producten die aan klantverwachtingen voldoen is goede communicatie tussen alle betrokkenen dus van essentieel belang. Als men deze factor onderschat kan dit zeer nadelige gevolgen hebben voor het eindresultaat van een ICT-project.

Het moge inmiddels duidelijk zijn dat om een goede requirements engineering te kunnen verrichten, er ook goede communicatie nodig is. Deze twee factoren zijn sterk aan elkaar gerelateerd. Zo dient men om de requirements te kunnen achterhalen met alle betrokkenen te communiceren. De relatie tussen de factor "slechte requirements" en "gebrekkige communicatie" is een goed voorbeeld van het feit dat meerdere factoren verantwoordelijk kunnen zijn voor het doen slagen of falen van een project.

6.4 Zijn succesvolle projecten zeldzaam?

Uit onderzoek is gebleken [17] dat ongeveer een kwart van het totaal aantal in de Verenigde Staten gestarte ICT-projecten vroegtijdig gestopt wordt. Bijna de helft van alle projecten wordt wel afgemaakt, maar wordt later opgeleverd dan gepland, kosten meer dan begroot of voldoen niet aan de eisen van de toekomstige gebruiker. Slechts één derde deel van de ICT-projecten wordt op tijd, binnen budget en tegen afgesproken kwaliteit opgeleverd.

Of een ICT-project succesvol is, hangt in grote mate vanaf in hoeverre het eindresultaat aan alle succescriteria voldoet. In het meest gunstige geval wordt er natuurlijk aan alle succescriteria voldaan, maar zoals de statistieken het weergeven is dit bij de meeste ICT-projecten helaas niet het geval. Toch is het achteraf niet altijd eenvoudig (zelfs wanneer een ICT-project niet aan alle succescriteria heeft voldaan) om deze als volledig succesvol of mislukt te kenmerken. Stel dat het opgeleverde systeem niet aan alle requirements heeft voldaan, maar de klant wel tevreden is over het eindresultaat. Is het project dan succesvol? Officieel gesproken niet, maar het zou niet ondenkbaar zijn als het project toch als geslaagd gezien wordt. Het systeem wordt immers voor de klant gemaakt en als deze tevreden is, is dat soms reden genoeg om het project toch als succesvol te beschouwen.

We hebben geleerd dat één van de belangrijkste redenen waarom ICT-projecten moeilijk te managen zijn, komt doordat het aantal aspecten, dat van invloed kunnen zijn op het resultaat van een ICT-project zo goed als onbeperkt zijn. Er bestaat namelijk geen alles overtreffende factor voor het doen slagen of falen van een project.

Als men puur naar de statistieken [16, 25] kijkt, lijkt het meer voor de handliggend dat een project eerder faalt dan slaagt. Dit hoeft echter niet te betekenen dat succesvolle projecten zeldzaam zijn, maar het dient meer als een waarschuwing gezien te worden, dat het moeilijker is dan het lijkt om ICT-projecten goed te managen. Wanneer een projectmanager gedurende het project niet de volledige aandacht erbij houdt, kan het dus gauw mis gaan.

6.5 Terugblik op dit hoofdstuk

De voornaamste reden waarom veel ICT-projecten door opdrachtgevers niet als succesvol worden ervaren komt doordat opgeleverde systemen niet aan de (klant) verwachtingen voldoen. Een slechte (incomplete) requirements engineering en/of een gebrekkige communicatie tussen betrokkenen ligt vaak ten grondslag voor het niet voldoen aan klantverwachtingen.

Als men ervoor wil zorgen dat systemen beter aan de klantverwachtingen voldoen, is het dan ook belangrijk om de requirements engineering en communicatie tussen de betrokkenen op één of ander manier te verbeteren. In de volgende hoofdstukken moet blijken of het gebruik van viewpoints, bijdraagt tot een verbetering van de requirements engineering en communicatie tussen betrokkenen. Als dit wel het geval blijkt te zijn, dan zal dit ook zeer waarschijnlijk een gunstig effect hebben op de doelstelling om producten beter aan klantverwachtingen te laten voldoen.

7. Viewpoints

Het ontwikkelen van systemen aan de hand van viewpoint-gebaseerde systeemontwikkelingmethoden is momenteel nog verre van gestandaardiseerd. Zo zijn er verschillende beschrijvingen van het begrip viewpoint in de omloop en hanteren diverse wetenschappers of wetenschappelijke organisaties (in meer of mindere mate) een eigen werkwijze op dit gebied. Soms kan men door de bomen het bos haast niet meer zien.

Om te kunnen analyseren of het werken met viewpoints uiteindelijk tot een betere requirements engineering en/of een betere communicatie tussen betrokkenen leidt, dient men dus eerst het begrip viewpoint te verduidelijken.

7.1 Wat zijn viewpoints?

Aan het begin van dit hoofdstuk werd aangeven dat er verschillende beschrijvingen voor het begrip "viewpoint" in de omloop zijn. Het is echter heel moeilijk om te zeggen welke beschrijvingen goed of fout zijn, aangezien er geen doorslaggevende autoriteiten zijn op dit gebied.

Gelukkig verschillen de meeste viewpoint beschrijvingen inhoudelijk gezien niet veel van elkaar en is het hierdoor mogelijk ze met elkaar te vergelijken. De beschrijving dat in IEEE 1471 [15] wordt gehanteerd vind ik persoonlijk het meest compleet en deze zal ik gedurende dit onderzoek blijven gebruiken.

IEEE 1471 beschrijving van het begrip viewpoint:

> "*A specification of the conventions for constructing and using a view. A pattern or template form which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.*" [15]

Een kale beschrijving op zichzelf alleen is zeker niet voldoende om een antwoord op de bovenstaande vraagstelling te kunnen verkrijgen. Daarom zal ik in de volgende paragrafen aan de hand van een aantal voorbeelden proberen uit te leggen wat viewpoints zijn en wat men ermee kan doen.

Om een completer beeld over viewpoints te geven, zal ik naast de literatuur uit IEEE 1471 ook literatuur uit andere studies gebruiken. Op deze manier creëer ik meer inzicht in de overeenkomsten en verschillen tussen verschillende viewpoint-gebaseerde systeemontwikkelingstheorieën.

7.1.1 Het meerdere perspectieven probleem

Bij de ontwikkeling van voornamelijk grote en complexe computersystemen zijn er vaak grote groepen mensen betrokken die ieders een uniek perspectief op te het ontwikkelen systeem hebben. Deze perspectieven worden onder andere aan de hand van hun kennis, expertise, vaardigheden, verantwoordelijkheden en belangen gevormd. Hierdoor kunnen perspectieven per persoon verschillend zijn. Neem bijvoorbeeld een traditioneel (systeemontwikkeling) projectgroep. Zo'n groep bestaat meestal uit meerdere mensen met

verschillende expertises en achtergronden. Iedere projectmedewerker heeft een rol of taak binnen de projectgroep en zal aan de hand van zijn of haar expertise het te ontwikkelen systeem vanuit een bepaald perspectief bekijken.

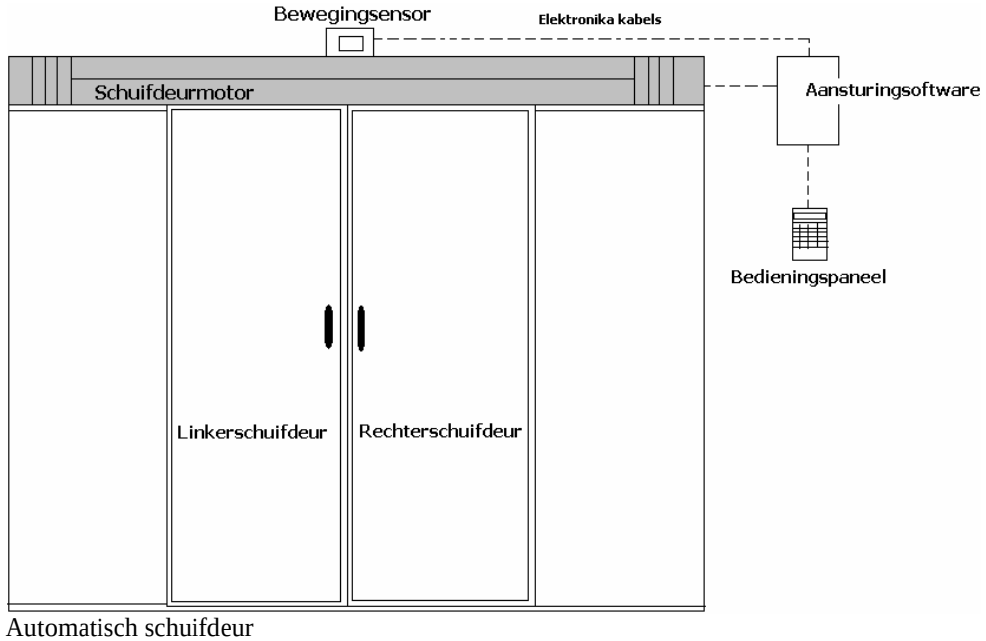
Zo zal een procesanalist het te ontwikkelen systeem vanuit een procesmatige perspectief bekijken en zal hij of zij aan de hand van procesgeoriënteerde modelleertechnieken (zoals dataflow- en activiteiten diagrammen) de systeemprocessen in kaart brengen of ontwerpen. Een data-analist daarentegen, zal vooral in het data gerelateerde aspect van het systeem geïnteresseerd zijn en zal aan de hand van datageoriënteerde modelleertechnieken (zoals FCO-IM) de informatiestromen in kaart brengen of ontwerpen. De gehanteerde modelleertalen en ontwikkelingsstrategieën zijn dus afhankelijk van de expertise van de persoon in kwestie.

Verschillende mensen uit verschillende expertises leveren dus niet alleen verschillende perspectieven op, maar kunnen daarnaast ook resulteren in het gebruik van verschillende modelleertechnieken en ontwikkelingsmethodes.

Hoe groter het aantal perspectieven waarmee rekening gehouden dient te worden, des te ingewikkelder het voor systeemontwikkelaars wordt om met elkaar te kunnen communiceren, de werkzaamheden te coördineren en een resultaat op te leveren dat aan de klantverwachtingen voldoet. Dit fenomeen staat ook wel bekend als het meerdere perspectieven probleem [28] en is iets waar ICT-projectmanagers, systeemontwikkelaars en IT-architecten bij grote en complexe projecten ongetwijfeld mee te maken zullen krijgen.

7.1.2 Meerdere perspectieven - een voorbeeld

Aan de hand van een relatief eenvoudig voorbeeld zal ik meer inzicht proberen te geven in het meerdere perspectieven probleem. Hieronder bevindt zich een schematische weergave van een nog te ontwikkelen automatische schuifdeur van een supermarkt ingang.



Figuur 1. Automatisch schuifdeur

Beschrijving van de onderdelen

Aansturingsoftware: De aansturingsoftware zorgt ervoor dat alle elektronische onderdelen met elkaar kunnen communiceren. Daarnaast is dit onderdeel ook verantwoordelijk voor de aansturing van de schuifdeurmotor.

Bewegingsensor: Wanneer een willekeurig persoon binnen het bereik van deze sensor komt, geeft het automatisch een elektronisch signaal aan de aansturingsoftware door.

Schuifdeurmotor: Zodra er een open- of dichtsignaal wordt ontvangen van de aansturingsoftware gaat de schuifdeurmotor draaien, waardoor de deuren respectievelijk open of dicht gaan.

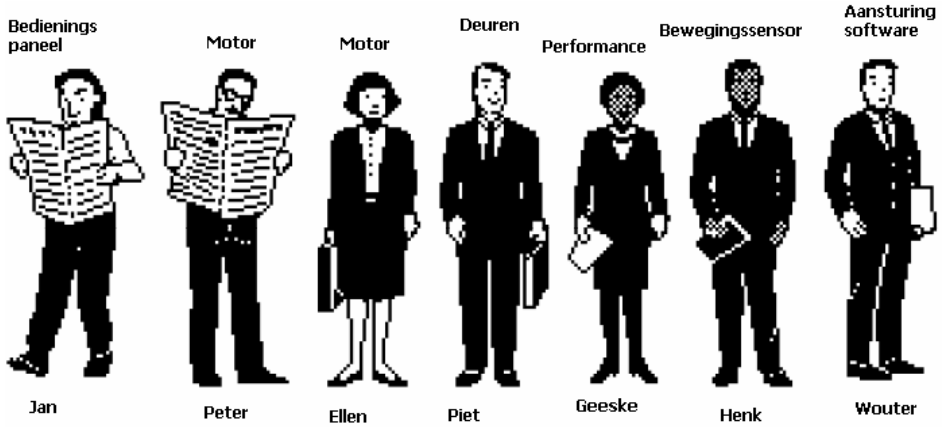
Bedieningspaneel: Via dit paneel kan de gebruiker het systeem aan of uit (dus de deuren open of op dicht) zetten. Ook kan men via dit paneel verschillende deurstanden selecteren Zo kan men ervoor zorgen dat de deur alleen nog maar van binnenuit open gaat.

Schuifdeuren: De schuifdeuren worden door de schuifdeurmotor aangedreven. Afhankelijk van de motorstand worden de deuren respectievelijk naar links of naar rechts geschoven.

Elektronicakabels: De elektronicakabels verbinden de verschillende onderdelen met elkaar en zorgen ervoor dat stroom- of data-impulsen van en naar de diverse onderdelen toe kunnen.

De projectgroep voor de realisatie van de schuifdeuren

Deze bestaat uit de volgende projectmedewerkers; Jan is verantwoordelijk voor het bedieningspaneel, Peter en Ellen zijn verantwoordelijk voor het ontwerpen van de schuifdeurmotor die de deuren openend en sluit, Piet is verantwoordelijk voor de deuren, Geeske voor performance (snelheid, kwaliteit, etc.) van het systeem (schuifdeuren) in het geheel, Henk ontwerpt de bewegingssensor en Wouter is verantwoordelijk voor de aansturingsoftware.



Figuur 2: De projectgroep met hun bijbehorende verantwoordelijkheden. [28]

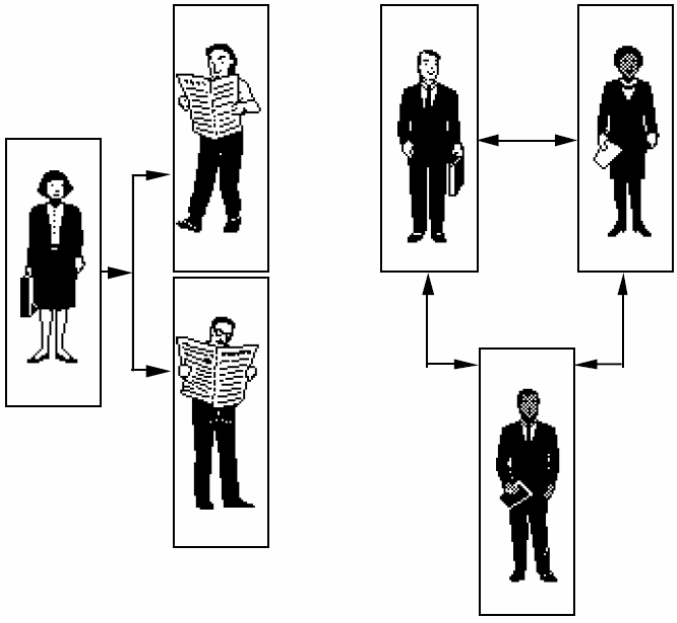
Specifieke aandachtspunten

Het werken met meerdere mensen en dus ook met meerdere perspectieven is niet eenvoudig. Om het meerdere perspectieven probleem verder te verduidelijken, zullen hieronder een aantal concrete voorbeelden gegeven worden van typische problemen die zich voordoen, wanneer er met meerdere perspectieven in een project wordt gewerkt.

Gedurende de ontwikkeling van het systeem doet een projectmanager er verstandig aan met deze aandachtspunten rekening te houden. Indien dit niet wordt gedaan, kunnen deze factoren de voortgang van een project ernstig belemmeren.

- Communicatie & relaties tussen actoren

  Goede communicatie tussen actoren is natuurlijk van essentieel belang voor het slagen van een project. Zo dient Wouter voor het correct functioneren van de aansturingsoftware met Peter, Ellen (schuifdeurmotor), Henk (bewegingsensor) en Jan (bedieningspaneel) te communiceren. Om de schuifdeuren open of dicht te kunnen laten gaan, dienen Peter en Ellen op hun beurt weer met Piet te communiceren. Peter en Ellen werken beiden aan de schuifdeurmotor, naast het communiceren naar Wouter (aansturingsoftware) en Piet (deuren), dienen ze onderling ook met elkaar te communiceren. Geeske moet uiteindelijk met iedereen communiceren om de schuifdeur op performance en kwaliteit te kunnen controleren.

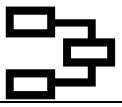
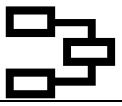
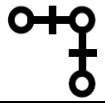
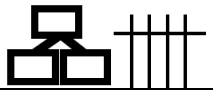


Figuur 3: Relaties tussen projectleden/actoren [28]

Er werken dus meerdere mensen aan verschillende onderdelen. Sommige projectmedewerkers moeten voordat ze verder kunnen met hun werkzaamheden, eerst op andere collega's wachten totdat zij klaar zijn met hun onderdeel. Zo kan de aansturingsoftware niet zonder een bedieningspaneel werken. Deze dient dus als eerst te worden gemaakt. Dit betekent dat Wouter op Jan moeten wachten voordat hij aan de slag kan. Actoren kunnen indirect een relatie met elkaar hebben, hierdoor kunnen in sommige gevallen bepaalde werkzaamheden niet parallel verricht worden. Men moet zich voorstellen dat hoe meer medewerkers en

perspectieven er zijn, des te moeilijker het wordt om met elkaar goed te kunnen communiceren.

- Verschillende modelleertechnieken
  Het systeem (de automatische schuifdeuren) bestaat uit meerdere onderdelen, waarbij ieder onderdeel z'n eigen functie heeft. Afhankelijk van de onderdeelfunctie en de expertise van de desbetreffende projectmedewerker dient er voor ieder onderdeel een geschikte modelleertaal of techniek gekozen te worden. Zo kan het gebeuren dat er binnen een projectgroep door verscheidene medewerkers, hoewel ze aan hetzelfde eindproduct werken, verschillende modelleertalen worden gehanteerd. In het voorbeeld gaan we er vanuit dat de projectmedewerkers voor het ontwerpen van de onderdelen, verschillende modelleertechnieken gebruiken.

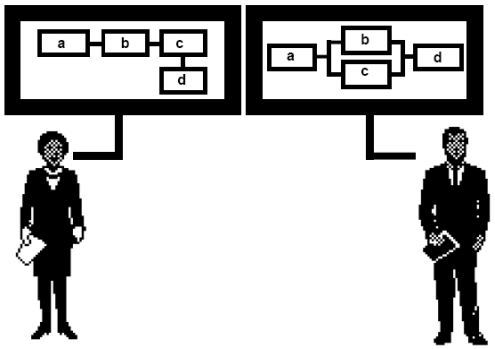
Medewerker	Onderdeel	Modelleertechniek	Voorbeeld
Jan	Bedieningspaneel	Functional Hierarchy	
Peter	Schuifdeurmotor	System Block Diagram	
Ellen	Schuifdeurmotor	System Block Diagram	
Piet	Schuifdeuren	Functional Hierarchy	
Geeske	Performance	Petri Nets	
Henk	Bewegingsensor	Object Structure	
Wouter	Aansturingsoftware	Object Structure & Action Tables	

De figuren die zijn gebruikt komen uit [28]

Het hebben van verschillende modelleertalen of -technieken binnen een projectgroep zal de communicatie tussen de projectmedewerkers onderling niet echt ten goed komen. Om in zulke gevallen elkaar toch te kunnen begrijpen dienen alle projectmedewerkers goed op de hoogte te zijn van de modelleertechnieken die gebruikt worden.

In de praktijk zal dit echter heel niet eenvoudig te realiseren zijn, dat projectmedewerkers alle modelleertechnieken zullen kennen. Vaak zijn medewerkers namelijk alleen gespecialiseerd in de modelleertechnieken die behoren tot hun expertise.

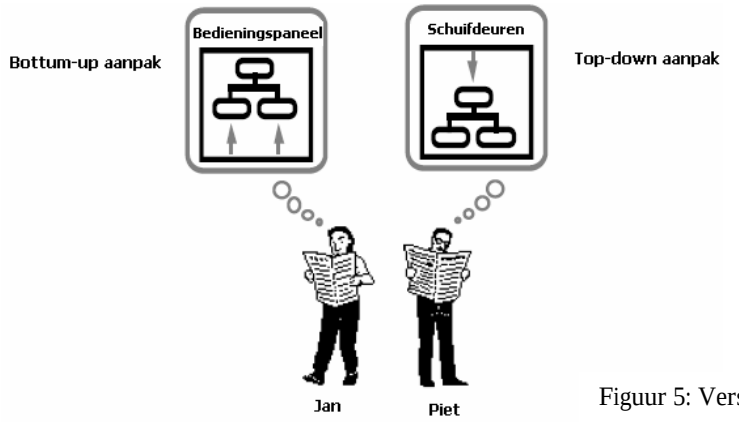
- Verschillende resultaten met dezelfde modelleertechniek
  Wanneer er met meerdere perspectieven gewerkt wordt, kan dit ook tot situaties leiden waarbij projectmedewerkers (hoewel ze dezelfde modelleertechnieken gebruiken) met verschillende ontwerpen op proppen komen. Als voorbeeld kunnen we Ellen en Peter nemen; beide personen werken aan de ontwikkeling van de schuifdeurmotor, gebruiken dezelfde modelleertechnieken, maar hebben beiden uiteindelijk een verschillend ontwerp voor de motor opgeleverd.



Figuur 4: Verschillende ontwerpen met zelfde modelleertechniek [28]

Men moet zich voorstellen dat wanneer er sprake is van verschillende ontwerpen voor dezelfde onderdelen, dit tot confrontaties en discussies kan leiden. Zo moeten nu niet alleen de verschillende ontwerpen met elkaar vergeleken worden, maar er moeten ook nog eens keuzes worden gemaakt. Vraagstukken omtrent de ontwerpkeuze zullen beantwoord moeten worden. Hoe men het wendt of keert, dergelijke scenario's als deze maken zowel de communicatie- als het ontwikkelproces gedurende het project complexer. Daarnaast vergen deze extra handelingen op hun beurt weer kostbare tijd en resources.

- Verschillende aanpakken met dezelfde modelleertechniek
  We hebben geleerd dat wanneer mensen dezelfde modelleertechnieken gebruiken, dit niet automatisch hoeft te betekenen dat ze hetzelfde resultaat zullen opleveren. Meerdere perspectieven kunnen er namelijk ook tot leiden, dat er per projectmedewerker verschillende werkmethoden en strategieën gebruikt kunnen worden. Hoewel Jan (bedieningspaneel) en Piet (schuifdeuren) dezelfde modelleertalen gebruiken, hanteren ze beiden een andere werkstrategie. Zo werkt Jan bottom-up en Piet op een top-down wijze.



Figuur 5: Verschillende strategieën [28]

> Hoe meer (verschillende) werkwijzen er binnen de projectgroep gehanteerd worden, des te moeilijker het wordt om een project goed te managen, aangezien iedere werkstrategie zijn eigen managementwijze vereist.

De bovenstaande aspecten vormen maar een kleine greep van de aspecten waar men als projectmanager rekening mee dient te houden wanneer men met meerdere perspectieven te maken heeft. Ook heb ik om het meerdere perspectieven probleem overzichtelijk te houden, opzettelijk geen actoren buiten de projectgroep in het voorbeeld geïntroduceerd. Zo is de actor "gebruiker" in het voorbeeld erbuiten gelaten.

Het moge duidelijk zijn dat bij grote en complexere projecten, het meerdere perspectieven probleem ongetwijfeld veel nadrukkelijker aanwezig zal zijn, dan dat het bij dit eenvoudige voorbeeld het geval was.

7.1.3 Viewpoint, de basisprincipes

Het werken aan de hand van viewpoints is een manier om stakeholder perspectieven beter te kunnen structuren, organiseren en te managen [28, 29]. Viewpoints kunnen dus gezien worden als een hulpmiddel wat het mogelijk maakt om met het meerdere perspectieven probleem om te kunnen gaan. Aan de hand van kleine voorbeelden zal hieronder het begrip viewpoint verder geconcretiseerd worden.

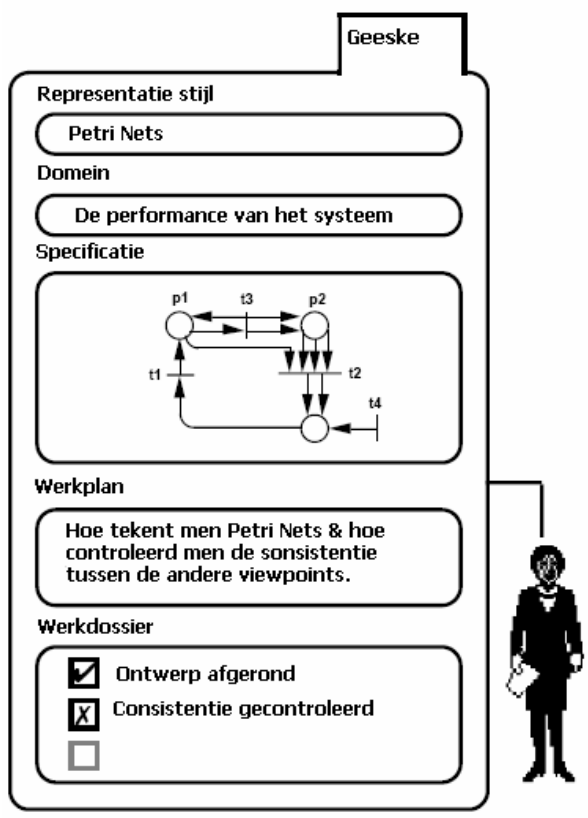
Viewpoint componenten

Volgens één van de eerste onderzoeken [28] op het gebied van viewpoint-gebaseerde systeemontwikkeling bestaat een viewpoint uit een vijftal componenten. Deze componenten worden ook wel slots genoemd.

Component / Slot	Beschrijving
Representatie stijl	Hier wordt de modelleertechniek vermeld, waarmee de view gemodelleerd dient te worden. Afhankelijk van het te modelleren domein, wordt er een geschikte modelleertaal gekozen.
Domein	Hier bevindt zich een (korte) tekstuele beschrijving van het te modelleren domein.
Specificatie	Dit slot bevat een uitwerking (model) van het domein doormiddel van de gekozen representatie stijl.
Werk plan	Hier wordt de methodiek (of het proces) beschreven hoe men tot de specificatie komt. Het is dus een soort handleiding om het model te kunnen opzetten.
Werkdossier	Hier bevindt zich een historisch overzicht van de verrichtte werkzaamheden en huidige stand van zaken omtrent de ontwikkelingsvoorgang. Het is als het ware een soort logboek.

Overigs is het handig om te weten dat de opmaak van een viewpoint in IEEE 1471 enigszins afwijkt van de hierboven gegeven beschrijving. Zo worden "slots" in IEEE 1471 "attributen" genoemd en verschillen deze inhoudelijk gezien ook enigszins van elkaar. Het attribuut "stakeholder" komt bijvoorbeeld alleen in IEEE 1471 voor. IEEE 1471 is voor een deel gebaseerd op literatuur uit het verleden. Om het principe achter IEEE 1471 beter te kunnen begrijpen is het belangrijk om ook enige kennis te hebben

over viewpoint-theorieën van voor het IEEE 1471 tijdperk. In de volgende paragraaf zal ik IEEE 1471 uitgebreid behandelden.



Figuur 6: Voorbeeld Viewpoint [28]

Hiernaast staat een voorbeeld van een viewpoint dat door Geeske gehanteerd wordt. Geeske is verantwoordelijk voor de performance van de automatische schuifdeuren (het systeem).

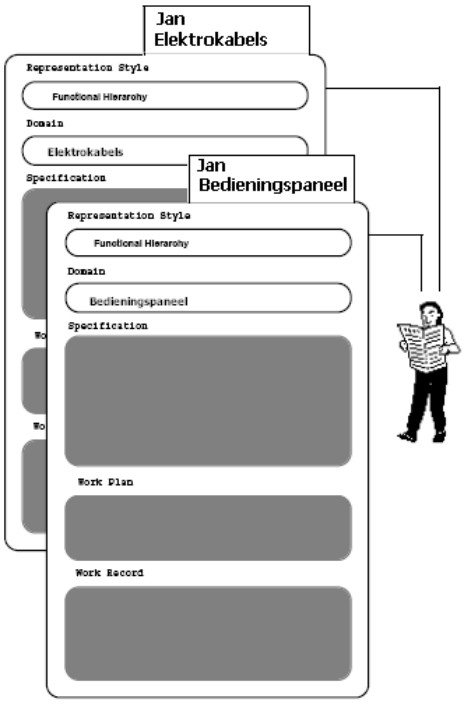
Als representatiestijl heeft ze voor Petri Nets gekozen. Dit heeft ze gedaan omdat dit het best past bij haar te analyseren domein; namelijk de performance van het systeem in het geheel.

Om haar doelstellingen te kunnen realiseren heeft ze een werkplan opzet waarin ze haar plan van aanpak beschrijft. Uiteindelijk kan ze aan de hand van dit werkplan haar uitwerking (specificatie) realiseren. Deze uitwerking wordt soms ook wel view genoemd.

In het werkdossier geeft ze aan hoever ze is gevorderd met haar werkzaamheden en welke handelingen inmiddels verricht zijn of nog verricht dienen te worden. Dit is een eenvoudige weergave van het geheel, maar men moet zich voorstellen dat bij grotere en complexere domeinen ook grotere en complexere viewpoints zijn.

Het kan ook gebeuren dat iemand meerdere domeinen tegelijkertijd analyseert, dus in andere woorden voor meerdere onderdelen tegelijkertijd verantwoordelijk is. Deze persoon beschikt in dit geval over meerdere viewpoints.

Een persoon kan dus meerdere viewpoints hebben. Het aantal viewpoints per persoon kan als het ware onbeperkt zijn. Zo is Jan naast het feit dat hij verantwoordelijk is voor het bedieningspaneel ook verantwoordelijk voor de elektronicakabels. Voor beiden viewpoints is het tevens mogelijk dat hij een andere representatie techniek gebruikt.

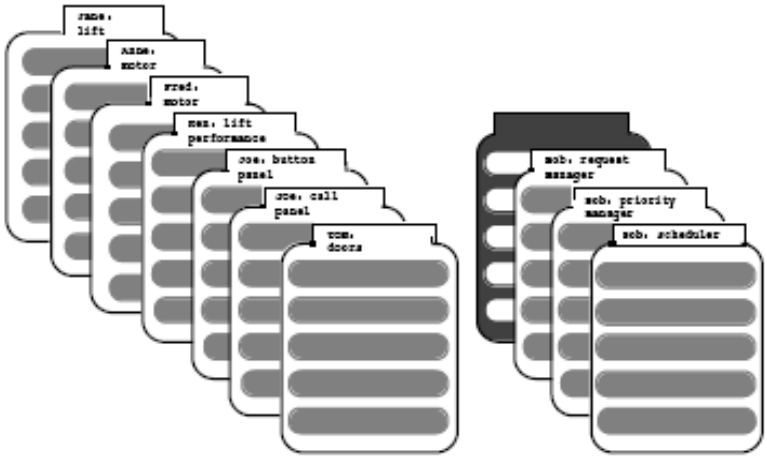


Figuur 7: Een persoon met één of meerdere viewpoints [28]

Mogelijkheden en risico's

- Wanneer uiteindelijk alle viewpoints in kaart zijn gebracht is het ook mogelijk ze met elkaar te vergelijken en ze doormiddel van consistentie-checks op eventuele fouten te controleren. Theoretisch gezien moet dit tot beter modelleren leiden. Ik zal hier later in het hoofdstuk op terugkomen.

- In dit eenvoudige voorbeeld met weinig mensen en perspectieven, is het werken met viewpoints nog relatief eenvoudig. Maar wanneer er meer mensen betrokken zijn en er sprake is van een complexer domein, loopt men gauw het risico het overzicht kwijt te raken.



Figuur 8: Veel viewpoints dragen niet bij tot de overzichtelijkheid.

Hoe meer viewpoints er zijn, des te moeilijker het is om het overzicht te kunnen behouden. Als men het werken aan de hand van viewpoint niet systematisch aanpakt, kan men het meerdere perspectieven probleem zelfs verergeren.

7.1.4 IEEE 1471

Een groot gedeelte van de informatie over viewpoints waar we momenteel over beschikken, stamt nog uit de begin jaren negentig. Zo is het voorbeeld met de automatische schuifdeuren gebaseerd op literatuur uit 1992 [28].

In het jaar 2000 heeft de IEEE een "recommended practice" omtrent het werken met viewpoints gepubliceerd. Deze "recommended practice" staat bekend als IEEE 1471 [15] en geniet momenteel een hoge mate van populariteit onder systeemontwikkelaars en IT-architecten die met viewpoints werken.

Mede dankzij IEEE 1471 is de bestaande theorie ten aanzien van het werken met viewpoints aanzienlijk uitgewerkt en verbeterd. Zo wordt er in IEEE 1471 aanbevolen om een domein (architectuur of systeem) te beschrijven in één of meerdere views. Een view wordt hierbij gedefinieerd als een "representatie van het gehele systeem vanuit een bepaald perspectief" [30]. De regels waar een bepaalde view aan moet voldoen wordt in IEEE 1471 een viewpoint genoemd.

Denkwijze
De basisprincipes voor het werken aan de hand van viewpoints zijn vastgelegd in IEEE 1471. Hierin worden de begrippen “stakeholder” (belanghebbende), “concern” (belang), “viewpoint” (gezichtspunt) en “view” (beeld) beschreven. Een architectuur (of domein) bestaat uit een aantal views. Views zijn representaties van een architectuur vanuit het perspectief van een gerelateerde set concerns en viewpoints bepalen hoe een view er uit moet zien.

IEEE 1471 is een beknopte standaard. Zo wordt er niet beschreven hoe views geselecteerd moeten worden en welke manier ze gemaakt moeten worden. Daarom is de verleiding voor veel ontwikkelaars groot om een architectuur raamwerk (view model) te adopteren. IEEE 1471 vertelt ons namelijk niet welke viewpoints belangrijk of nuttig zijn. Hiervoor kunnen we view modellen gebruiken die bijvoorbeeld door Zachman [40], Kruchten [31] en Soni [32] geïntroduceerd zijn.

Het uitgangspunt achter IEEE 1471 is echter dat architectuur- of domeindocumentatie wordt opgesteld vanuit de gedachtewereld van stakeholders. “Het is de vraag of een raamwerk dat opdelingen op allerlei logische gronden scheidt (statisch van dynamisch, functioneel van implementatie) overeenkomt met de informatiebehoefte van een bepaalde doelgroep. Bovendien lijken de raamwerken met hun focus op de beschrijving van het systeem zelf, zich nauwelijks te bekommeren over allerlei andere belangrijke vragen, zoals vraagstukken die te maken hebben met mogelijke risico’s, kosten en keuzes die gemaakt moeten worden op basis van principes” [54].

Kenmerken [55]
Formeel gezien heet IEEE 1471, ANSI/IEEE 1471-2000, *Recommended Practice for Architecture Description of Software-Intensive Systems.*

- Het biedt naast definities ook een metamodel aan om architecturen (of domeinen) te beschrijven;

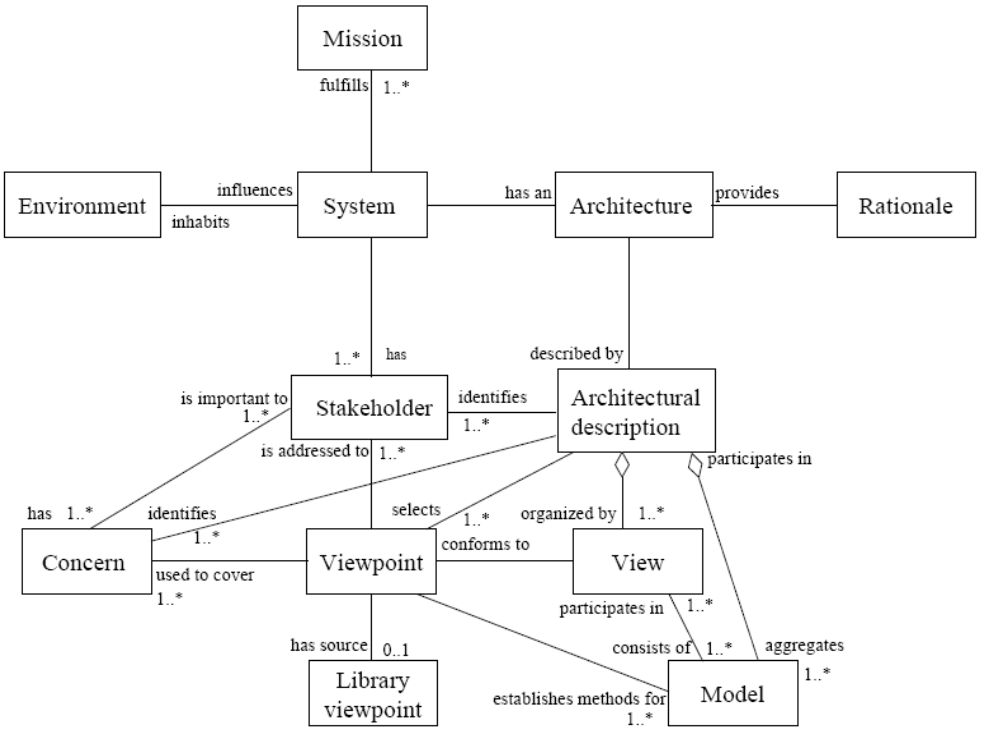
- Het gaat er vanuit dat architectuurbeschrijvingen van systemen aan stakeholder concerns dienen te voldoen;

- Het gaat er vanuit dat architecturen altijd vanuit meerdere perspectief bekeken dienen te worden. Een enkele view is niet voldoende om alle stakeholder concerns te bevatten;

- Het is een beperkte standaard (vastgesteld op basis van consensus) met een breed draagvlak.. Zo doet het geen uitspraken over de beschreven systemen, maar alleen over de beschrijvingen. Hiernaast worden er geen processen of tools gedefinieerd die de ontwikkelaar ondersteunen bij het verrichten van z’n werkzaamheden;

- Het heeft vooral betrekking op (enterprise) systemen waarin software een essentiële invloed op de gehele organisatie heeft.

Metamodel IEEE 1471



Figuur 9: Metamodel IEEE 1471 [15]

Toelichting metamodel

Een systeem wordt gedefinieerd als een verzameling van componenten die als doelstelling heeft een specifieke functie of een set van functies te vervullen. Hierdoor kan het begrip "systeem" behoorlijk breed geïnterpreteerd worden. Wat in feite betekent dat bijna alles (mits het geheel een bepaalde doelstelling of functie vervult) als een systeem beschouwd kan worden.

Een systeem bevindt zich altijd in een omgeving (environment). Deze omgeving kan het systeem op diverse manieren (bv. politiek, operationeel, wetgeving, etc.) beïnvloeden. De omgeving bepaald voor een groot gedeelte de setting en de omstandigheden waarin het systeem uiteindelijk ontwikkelt dient te worden. Naast het beïnvloeden van systemen kan een omgeving ook systemen aan elkaar (direct of indirect) koppelen. In feite bepaalt de omgeving de scope van het systeem. De grens tussen het systeem en de omgeving wordt ook wel interface genoemd.

Een systeem heeft één of meerdere stakeholders (belanghebbenden), waarbij iedere stakeholders één of meerdere concerns (belangen) ten aanzien van het systeem heeft. Concerns zijn interessegebieden die te maken kunnen hebben met de ontwikkeling, de werking en andere aspecten van het systeem die door stakeholders als belangrijk worden ervaren. Typische voorbeelden van stakeholder concerns zijn functionaliteit, performance, beveiliging, betrouwbaarheid, kosten, veiligheid, etc. Ik zal hier verderop in het hoofdstuk op terugkomen.

Een systeem is ontwikkeld om één of meerdere missies (missions) in zijn omgeving te realiseren. Ieder systeem heeft zijn bestaansrecht te danken aan een doelstelling. Een systeem dient één of meer doeltellingen te vervullen van één of meer stakeholders.

Voor de introductie van IEEE 1471 was de terminologie voornamelijk gebaseerd op het systeem en z'n omgeving. Het grootste gedeelte van IEEE 1471 is echter gefocust op het beschrijven van architecturen. Zo worden er in IEEE 1471 een aantal nieuwe termen en begrippen gepresenteerd die gerelateerd zijn aan architecturen.

Iedere systeem heeft een architectuur dat door middel van een architectuurbeschrijving (architectural description) beschreven kan worden. Houdt er wel rekening mee dat een architectuur van een systeem (wat conceptioneel is) niet hetzelfde is als een architectuurbeschrijving (wat concreet is). Een architectuurbeschrijving is in IEEE 1471 gedefinieerd als een verzameling van producten om een architectuur te documenteren.

Een architectuurbeschrijving kan in één of meerdere views worden verdeeld. Iedere view dekt één of meerdere stakeholderbelangen. Een view wordt gedefinieerd als een representatie van een systeem vanuit het perspectief van een gerelateerde set van belangen. Een view wordt gecreëerd aan de hand van regels en conventies die in een viewpoint zijn gedefinieerd. Een viewpoint wordt gedefinieerd als een specificatie van regels om een view te maken en te kunnen gebruiken. Het is als het ware een sjabloon om individuele views te ontwikkelen. Dit wordt onder andere gedaan door het vastleggen van de viewpointdoelstelling, publiek (doelgroep) voor de view en de techniek om deze views te creëren en te analyseren.

Hieronder wordt een korte samenvatting gegeven van de begrippen die binnen IEEE 1471 gehanteerd worden. Omdat Engelse en Nederlandse termen in dit onderzoek vaak door elkaar heen gebruikt worden, is er ter verduidelijking per begrip (indien aanwezig) een Nederlandse vertaling toegevoegd.

Begrip	Vertaling	Beschrijving
System	Systeem	Het systeem waarvan de architectuur beschreven zal worden.
Environment:	Omgeving	De omgeving waarin het systeem zal functioneren en die het systeem zal beïnvloeden.
Mission	Missie	Doelstelling van het systeem binnen de omgeving.
Architecture	Architectuur	De architectuur van het systeem.
Architecture Description	Architectuurbeschrijving	Verzameling van documenten die samen de architectuur van het systeem beschrijven.
Stakeholder	Belanghebber	Een persoon of organisatie (actor) die een belang heeft bij het systeem.
Concern	Belang	Belang (of zorg) van een stakeholder.
View	Beeld	Is een onderdeel van de architectuurbeschrijving.

Viewpoint	Viewpoint	Voorschrift voor een view (inhoud en de te gebruiken modellen, relatie naar concerns en stakeholders).
Model	Model	Een stuk gestructureerde informatie met een bepaalde representatie en eventueel een instructie voor het opstellen van de representatie.
Rationale	Reden	Uitleg over de architectuurbeschrijving (in het bijzonder welke viewpoints er zijn gekozen en waarom).
Library Viewpoint	Library Viewpoint	Een viewpoint wat is vastgelegd voor het gebruik in meerdere architectuurbeschrijvingen.

Views, Viewpoints en Stakeholders

Een view is een weergave van het systeem vanuit het perspectief van een set belangen behorende bij één of meerdere stakeholders. In een viewpoint wordt beschreven hoe een view gemaakt moet worden. Heel grof samengevat is een view wat je ziet en is een viewpoint de positie waar vanuit je het bekijkt [50].

Viewpoints zijn bedoeld als een hulpmiddel om te kunnen focussen op bepaalde aspecten van een architectuur. Deze aspecten worden bepaald door de concerns van een stakeholder. Wat wel en niet zichtbaar moet zijn vanuit een bepaald viewpoint is dus in grote mate afhankelijk van de concern(s) van een stakeholder. Om het begrip "stakeholder" verder te verduidelijken, staan hieronder een aantal voorbeelden van diverse stakeholders samen met hun concerns.

Stakeholder	Voorbeeld concern
Eindgebruiker	Wat zijn de verwachtte consequenties voor de werkzaamheden en werkplek?
IT-architect	Wat zijn de consequenties voor onderhoudbaarheid van het systeem op het gebied van correctief-, preventief- en adaptief-onderhoud?
Directie	Hoe kan men ervoor zorgen dat het beleid zal worden gevolgd, zowel tijdens de ontwikkeling als gedurende het gebruik van het systeem? Welke gevolgen zullen beslissingen hebben op het gebied van personeel, ICT en financiën?
Systeemonderhoud manager	Voor welke nieuwe technieken moeten er voorbereidingen getroffen worden. Moet het huidige onderhoudsproces worden aangepast? Wat voor gevolgen heeft het voor bestaande applicaties? Hoe moeten de systemen beveiligd worden?
Projectmanager	In hoeverre zullen de huidige werkprocessen afhankelijk zijn van het nog te ontwikkelen systeem? Hoe kan het project succesvol gemanaged worden?

Concerns vormen tevens ook de basis voor het opzetten van een stakeholder-profielmodel. Verderop in dit hoofdstuk zal worden uitgelegd hoe men tot een dergelijk model komt.

Viewpoints zijn bedoeld om over bepaalde delen van een architectuur te kunnen communiceren. De communicatie tussen stakeholders en IT-architecten kan zowel puur informatief (dus eenrichtingsverkeer) als tweerichtingsverkeer van karakter zijn. Over het algemeen komt de laatste vorm van communicatie meer voor. Zo zal de IT-architect de stakeholder infomeren en de stakeholder zal vervolgens zijn of haar feedback op de door de IT-architect gepresenteerde aspecten geven.

Wat wel en niet wordt weergegeven in een view is dus in grote mate afhankelijk van de scope van een viewpoint en wat relevant is voor stakeholder concerns. Vaak zijn deze twee aspecten (min of meer) hetzelfde. Dit komt omdat viewpoints aan de hand van stakeholder concerns worden ontworpen. De mate in hoeverre een stakeholder concern relevant is, wordt daarom ook als een van de belangrijkste selectiecriteria gezien bij het bepalen wat wel en wat niet in een view dient worden te weergeven.

Naast de informatie die in een view dient worden te weergegeven, kan een architectuurbeschrijving tevens ook nog andere informatie bevatten. Hierbij kan men denken aan zaken zoals de gedachtegoed achter het systeem en het bieden van een systeemoverzicht. Deze additionele informatie valt binnen IEEE 1471 officieel niet onder de definitie van een viewpoint, maar wordt voor organisatorische redenen wel aangeraden te vermelden.

In een architectuurbeschrijving worden één of meerdere viewpoints geselecteerd. Aan van deze viewpoints worden vervolgens de gerelateerde views gemaakt. De keuze van welke viewpoints er geselecteerd dienen te worden, hangt in grote mate af van de stakeholder concerns die van belang zijn voor de desbetreffende architectuurbeschrijving. Het ISO Reference Model of Open Distributed Processing (RM-ODP) [33] kent bijvoorbeeld vijf viewpoints. In IEEE 1471 daarentegen worden er geen viewpoints gedefinieerd of beschreven. Een viewpoint kan zowel aan de hand van een architectuurbeschrijving geselecteerd worden, maar deze kunnen ook ergens anders gedefinieerd worden. Extern gedefinieerde viewpoints worden binnen IEEE 1471 ook wel library viewpoints genoemd. De vijf viewpoints die binnen RD-ODP gebruikt worden zijn typische voorbeelden van library viewpoints. Later in dit hoofdstuk zal ik hierop teruggekomen.

Een view mag één of meerdere modellen bevatten. Ook mogen deze modellen in één of meerdere views gebruikt worden. Iedere model wordt gedefinieerd aan de hand van de methodiek die in de bijbehorende viewpoint-specificatie staat beschreven. Een architectuurbeschrijving bepaalt de modellen en organiseert deze in views.

Stakeholder-profielmodel

Om stakeholder concerns te kunnen achterhalen dient men natuurlijk eerst de stakeholders te identificeren. Dit kan aan de hand van een stakeholder-profielmodel worden gedaan. Het hebben van stakeholder-profielmodel is namelijk een belangrijk voorwaarde om een viewpoint-specificatie op te kunnen zetten.

Het in kaart brengen van de stakeholders en hun bijbehorende profielen kan doormiddel van de onderstaande sjabloon worden gedaan.

Attribuut	Betekenis
Titel	Korte herkenbare omschrijving van rol of functie.
Doelen	Het doel(en) van de rol of functie. Een doel is hier gezien worden als een conditie die bereikt of gehandhaafd moet worden.
Taken	Activiteiten die uitgevoerd moeten worden in het kader van de rol of functie.
Concepten	Objecten die van belang zijn voor de rol of functie en die samen het "wereldbeeld" bepalen.
Concerns	Concrete belangen of zorgen die richting geven aan de activiteiten en die bepalen welke diensten verlangd worden van andere rollen of functies.

Hoe meer stakeholders er zijn, des te meer profielenmodellen er gemaakt dienen te worden. Om het begrip "stakeholder-profielmodel" verder te concretiseren, staat hieronder een eenvoudig voorbeeld worden geven van het profielmodel van een IT-architect.

Attribuut	Betekenis
Titel	IT-architect
Doelen	Een beheerste, continue en efficiënte werking van de automatisering in een bepaald domein voor nu en in de toekomst.
Taken	Opstellen van architecturen. Uitdragen en bewaken van architecturen. Geven van projectadviezen.
Concepten	Bedrijfsfuncties (intern/extern), bedrijfsstrategie, IT-operaties, IT-strategie, applicaties, gegevens, computers, systeemsoftware, architecturen, toekomstverwachtingen.
Concerns	Hoe blijf ik geïnformeerd? Hoe maak ik de juiste keuzes? Hoe communiceer ik effectief? Welke diensten worden verlangd van andere rollen of functies?

7.1.5 Werkwijze binnen IEEE 1471

De werkwijze waarop men met viewpoints dient te werken wordt in IEEE 1471 slechts summier beschreven. Men gaat vooral in op de doelstelling ("wat") in plaats van de methodiek ("hoe"). Dit heeft natuurlijk wel z'n voor- en nadelen. Zo heeft een IT-architect hierdoor een hoge mate van vrijheid in de manier waarop hij of zij wil werken, maar aan de andere kant zal een onervaren IT-architect daarentegen veel meer moeite hebben om de werkzaamheden zonder een begeleidende methodiek naar wens te kunnen verrichten.

Voor wie is het bedoeld?

IEEE 1471 is bestemd voor stakeholders die te maken hebben met de ontwikkeling en evolutie van een systeem. Enkele typische stakeholders zijn:

- Personen die het systeem gebruiken, eigenaar van het systeem zijn of van plan zijn deze te kopen. Bijvoorbeeld: gebruikers, operatoren en kopers;
- Personen die architecturen ontwikkelen, leveren of documenteren. Bijvoorbeeld: IT-architecten;
- Personen die systemen ontwikkelen, leveren of onderhouden. Bijvoorbeeld: IT-architecten, ontwerpers, programmeurs, systeembeheerders, domein engineers, project managers, systeem ontwikkelaars en configuratie managers;
- Personen die de ontwikkeling van systemen overzien of evalueren. Bijvoorbeeld: Chief Information Officers (CIO), auditors en onafhankelijke reviewers.

Wat wordt er behandeld?
IEEE 1471 bevat geen uitgebreide tekstuele beschrijvingen waar het één en ander tot op het bot wordt uitgelegd en gedicteerd. Toch worden er een aantal zeer belangrijke onderwerpen ten aanzien van het werken met viewpoints behandeld. Zoals:

- Architectuurdocumentatie: Binnen IEEE 1471 wordt er alleen een beschrijving gegeven van welke informatie een architectuurdocumentatie minimaal dient te bevatten. Zoals:

  - Datum van uitgifte en documentstatus;
  - Organisatienaam;
  - Versie (verandering) historie;
  - Samenvatting;
  - Scope;
  - Context;
  - Beschrijving van de terminologie;
  - Referenties.

- Identificatie van stakeholders en concerns: IEEE 1471 bevat een overzicht van welke stakeholders er minimaal geïdentificeerd dienen te worden. Het betreft de volgende stakeholders:

  - Gebruikers van het systeem;
  - Kopers (opdrachtgever) van het systeem;
  - Ontwikkelaars van het systeem;
  - Beheerders van het systeem.

  Daarnaast bevat IEEE 1471 tevens ook een overzicht van welke concerns er minimaal geïdentificeerd dienen te concerns. Het betreft de volgende concerns:

  - De doelstelling of missie van het systeem;
  - Geschiktheid van het systeem om zijn doelstelling of missie te vervullen;
  - De mate van haalbaarheid om het systeem te ontwikkelen;
  - Welke risico's tijdens de ontwikkeling en na de realisatie van het systeem voor gebruikers, kopers en ontwikkelaars van het systeem kunnen ontstaan;
  - De mate van onderhoud-, implementeer- en evolueerbaarheid van het systeem.

- Keuze van (architectuur) viewpoints: Er wordt in IEEE 1471 alleen maar aangegeven dat er viewpoints geselecteerd dienen te worden. Helaas wordt er niet gezegd welke viewpoints dit zijn en hoe dit moet gebeuren. Het enige wat wordt beschreven, is dat een viewpoint uit de volgende componenten dient te bestaan:

  - Een viewpoint naam;
  - De stakeholders die door de viewpoint geadresseerd worden;
  - De concerns die door de viewpoint geadresseerd worden;
  - De modelleertaal of techniek die gebruik zal worden om de view te kunnen modelleren;
  - De bron.

- (Architectuur) views: Bij iedere view hoort precies één viewpoint. Daarnaast dient elke view te voldoen aan de viewpoint-specificatie die behoort tot de corresponderende viewpoint. Een view dient uit de volgende elementen te bestaan:

  - Tekstuele beschrijving of andere introducerende informatie;
  - Representatie van het systeem dat gemaakt is aan de hand van de modelleertaal of -techniek wat is voorgeschreven in de bijbehorende viewpoint;
  - Configuratie informatie.

- Consistentie tussen views: IEEE 1471 vereist dat er wordt vermeld of er inconsistenties tussen views bekend zijn. Daarnaast dient er een analyse verricht te worden over alle views om inconsistenties tussen views te kunnen achterhalen. De resultaten uit deze analyse dienen natuurlijk ook worden vermeld.

- Architectuur redenen: Iedere architectuurbeschrijving dient de argumentatie te vermelden voor de concepten die zijn geselecteerd en de keuzes die er zijn gemaakt. Ook moet er bewijs worden geleverd dat men ook aan andere alternatieven heeft gedacht en dient men te onderbouwen waarom men uiteindelijk toch voor een bepaalde optie heeft gekozen.

Wat wordt er vereist?
Om de volgens de IEEE 1471 principe te kunnen werken, dient men aan de onderstaande aspecten te voldoen:

- Stakeholders en hun concerns dienen geïdentificeerd te worden;
- Er dienen viewpoints geselecteerd te worden en de argumentatie achter die selectie dient vermeld te worden. Daarnaast dient er beschreven te worden welke stakeholders of concerns door deze viewpoints worden bediend;
- Een systeem dient in views te worden gerepresenteerd;
- Bekende inconsistenties tussen views dienen vermeld te worden;
- De argumentatie achter keuzes dienen vermeld te worden.

Wat wordt niet vereist?
Hoewel IEEE 1471 een aantal basiseisen kent, biedt het systeemontwikkelaars en IT-architecten, toch een hoge mate van vrijheid op de manier waarop ze deze eisen willen

realiseren. Deze vrijheid wordt mede verkregen vanwege het feit dat de onderstaande aspecten niet worden vereist.

- Een specifieke beschrijvingstaal of –techniek om views te kunnen beschrijven;
- Specifieke modellen om views te representeren;
- Formele consistentiecriteria om views op consistentie te kunnen controleren.

Waarom IEEE 1471?

Er zullen ongetwijfeld een veel redenen zijn waarom iemand IEEE 1471 zou moeten gebruiken. Om het overzichtelijk te houden worden hieronder alleen de belangrijkste redenen.

- Aan de hand van de eisen die in IEEE 1471 voor architectuurbeschrijvingen zijn voorgesteld, is het mogelijk de kwaliteit van een architectuurbeschrijvingen vastleggen of bepalen;
- Op basis van de gemaakte architectuurbeschrijvingen is men in staat om de kwaliteit van nog te ontwikkelen systemen (enigszins) te voorspellen;
- Omdat alle architectuurbeschrijvingen dezelfde structurele opbouw hebben, is het mogelijk deze onderling met elkaar te vergelijken. Op deze manier kunnen er fouten worden ontdekt en is men in staat om beter onderbouwde keuzes of beslissingen te maken.

Voorbeeld van een IEEE 1471 viewpoint

In IEEE 1471 is een viewpoint een kort overzicht (circa één pagina) met een beknopte uitleg over de naam van de viewpoint, beschrijving van de viewpoint, doelgroep voor wie de viewpoint bestemd is, de belangen die middels deze viewpoint behartigd worden, de taal en modelleringtechnieken die gebruikt moeten worden om de gerelateerde view te kunnen representeren. Een viewpoint wordt dus gespecificeerd aan de hand van een:

- Een viewpoint naam;
- De stakeholders die door de viewpoint geadresseerd worden;
- De stakeholder concerns die gerelateerd zijn aan de viewpoint;
- De analyse methoden, modelleertaal of -techniek, die gebruikt moeten worden om de view te kunnen representeren;
- De bron van de viewpoint (bv, auteur, literatuur, citaat).

Attribuut	Beschrijving
Viewpoint naam:	Capability Viewpoint
Stakeholders:	Cliënten, producers, ontwikkelaars en integrators
Concerns:	Op welke manier zijn de functionaliteiten van het systeem gebundeld? Welke interfaces worden er gemanaged?
Viewpoint taal:	De componenten en hun functies (UML diagrammen) De Interface en hun attributen (UML klasse diagrammen)
Bron:	Deze viewpoint wordt soms ook wel gekenmerkt als Static, Application- en Structural viewpoints

Voorbeeld komt uit [15]

7.1.5 Relatie van IEEE 1471 ten aanzien van andere theorieën

In paragraaf 7.1.3 werd het begrip viewpoint voor het eerst beschreven. Deze beschrijving was voornamelijk gebaseerd op literatuur uit de begin jaren negentig en verschilt daardoor enigszins van de viewpoint-specificatie beschrijving zoals die door IEEE 1471 gehanteerd wordt. Zo wordt een viewpoint volgens paragraaf 7.1.3 aan de hand van een vijftal componenten gevormd, namelijk: representatiestijl, domein, specificatie, werkplan en werkdossier. Binnen IEEE 1471 bestaat een viewpoint ook uit vijf componenten, maar deze verschillen inhoudelijk gezien van de viewpoint-specificatie beschrijving dat in paragraaf 7.1.3 wordt gehanteerd. Volgens IEEE 1471 bestaat een viewpoint namelijk uit de componenten: viewpoint naam, stakeholders, concerns, viewpoint taal en bron. Het feit dat beide viewpoints beschrijvingen van elkaar verschillen vormt geen probleem, maar het kan natuurlijk wel enige verwarring veroorzaken. Zolang het begrip viewpoint wereldwijd nog niet officieel door alle partijen is gestandaardiseerd, is het momenteel echter niet eenvoudig om te zeggen welke beschrijvingen goed of fout zijn.

Wanneer men een systeem vanuit een architecturaal oogpunt ontwikkelt, zou het gebruik van een IEEE 1471 viewpoint-specificatie beter tot zijn recht komen. Bij traditionele systeemontwikkeling daarentegen komt het gebruik van een viewpoint-specificatie zoals die in paragraaf 7.1.3 is beschreven beter tot z'n recht. Echter blijft het gedachtegoed achter het gebruik van viewpoint-gebaseerde systeemontwikkelingmethoden voor beide varianten hetzelfde [51]. Er wordt immers vanuit de concern van de stakeholder gewerkt en in beide alternatieven staat het werken met meerdere stakeholder perspectieven centraal. Voor het eindresultaat zal het dan ook niet veel uitmaken welke viewpoint-gebaseerde methodiek er uiteindelijk gebruikt worden.

Requirements Engineering en het gebruik van viewpoints

Wanneer men bij het verrichten van een requirements engineering een viewpoints georiënteerde werkmethode besluit te hanteren, erkent men eigenlijk indirect dat het niet mogelijk is om vanuit één perspectief alle systeem requirements te herkennen [36].

Viewpoints bevatten onder andere stakeholders en hun concerns. Afhankelijk van de viewpoint dat gekozen is, wordt de focus van de requirements engineer op concerns gelegd die door dat viewpoint worden geadresseerd. Aan de hand van een viewpoint-specificatie probeert de requirements engineer de requirements die voor dat viewpoint relevant zijn te achterhalen. Viewpoints hebben dus naast een beschrijvende eigenschap ook een sturende eigenschap. Ik zal hier in het volgend hoofdstuk op terugkomen.

7.2 In hoeverre kunnen viewpoints gemodelleerd worden?

Vanuit een praktisch oogpunt gezien heeft het niet veel zin om viewpoints te modelleren, aangezien een viewpoint een soort handleiding is om tot een view te kunnen komen. Het is veel interessanter om views te modelleren. Het was dan ook misschien beter geweest om de bovenstaande vraag te veranderen naar: "Kunnen views gemodelleerd worden?".

Views kunnen wel degelijk gemodelleerd worden [29]. Het zijn namelijk representaties van perspectieven. De vraag is alleen welke modelleertalen of -technieken zijn hiervoor het best geschikt? Het antwoord op deze vraag helaas niet eenvoudig te geven. Dit komt doordat de te gebruiken modelleertaal of -techniek in sterke mate afhankelijk is van de

viewpoint dat gehanteerd wordt. IEEE 1471 laat de modelleur in dit geval in het duister tasten. Afgezien van een paar zeer beknopte voorbeelden wordt er namelijk geen concrete methodiek beschreven over hoe en met welke technieken views gemodelleerd moeten worden.

Om views te kunnen modelleren kan men gebruik maken van zogenaamde view modellen (deze worden vaak architectuurraamwerken genoemd). Een view model bestaat uit een verzameling van voorgeschreven viewpoints (perspectieven) op een domein (architectuur of systeem) [30]. View modellen kunnen zowel een voorschrijvend als een beschrijvend karakter hebben. Zo kunnen ze onder andere voorschrijven aan welke aspecten er aandacht geschonken dient te worden tijdens de ontwikkeling van een architectuur. Hun beschrijvende rol is dat ze een raamwerk voor de documentatie van architecturen bieden.

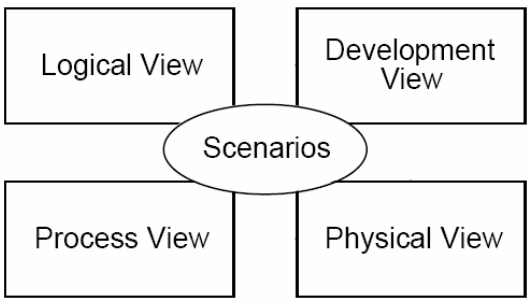
Er bestaan twee soorten architectuurraamwerken in de vorm van enterprise- en applicatieraamwerken [34]. Enterpriseraamwerken hebben de hele organisatie op het oog, bevatten vaak meerdere perspectieven (dimensies) en leiden tot veel modellen en documenten. Applicatieraamwerken zijn gericht op één applicatie of een reeks van vergelijkbare applicaties. Ze bevatten vaak een beperkt aantal modellen en leiden veelal tot maar één document. De informatie in applicatieraamwerken is vaak van een lager detailniveau dan de informatie die in enterpriseraamwerken weergeven wordt.

Om meer inzicht te verkrijgen in architectuurraamwerken, zullen hieronder een aantal bekende applicatie- en enterpriseraamwerken worden behandeld. Deze raamwerken zullen echter niet uitvoerig geanalyseerd en beschreven worden. Het gaat er alleen om dat men een beeld krijgt van de mogelijkheden. Het analyseren, vergelijken en verbeteren van de verschillende view modellen valt overigens buiten de scope van dit onderzoek.

7.2.1 Kruchten 4+1 view model

Philippe Kruchten, de bedenker van dit applicatieraamwerk ging er vanuit dat het onmogelijk is om vanuit één diagram (of perspectief) de essentie van een systeem in kaart te brengen. De meeste (enterprise)systemen en hun omgeving zijn daar simpelweg gewoon te complex voor. Om de essentie van een systeem te kunnen achterhalen, dient een systeem hierom vanuit meerdere invalshoeken te worden bekeken.

De 4+1 view model van Kruchten [31, 37], specificeert vier verschillende perspectieven op een systeem, namelijk: "*logical*", "*process*", "*development*" en "*physical*" perspectief. Deze perspectieven worden tenslotte middels een "*scenario's*" (de +1) perspectief met elkaar verbonden. Ieder perspectief richt zich op een specifiek set concerns.



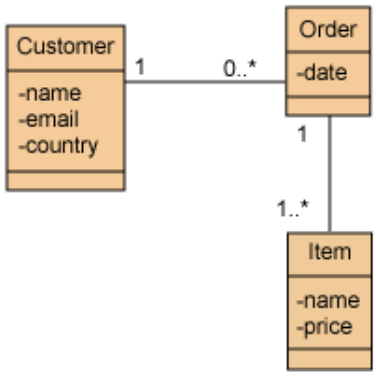
Figuur 10. Het 4 + 1 model [31]

Beschrijving van de views

Hieronder zullen de views op de vorige pagina, één voor één behandeld worden. Zo zal er per view worden beschreven wat deze inhoudt, met welke modelleertechnieken de view gemodelleerd kan worden en zal er tevens per view ook een klein voorbeeld gegeven worden van hoe het resultaat (model) van zo'n view eruit kan zien.

1. *Logical* view

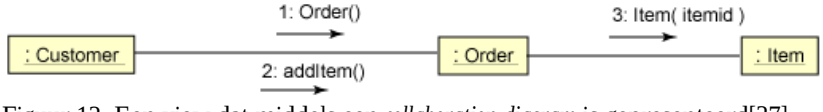
De logical view illustreert hoe het systeem in functionele onderdelen is opgedeeld. Klassen en objecten vormen de belangrijkste elementen binnen deze view. Doormiddel van klasse-, collaboration- en sequentie-diagrammen kunnen de relaties tussen de elementen weergegeven worden. Wanneer een applicatie erg datagericht is, kan er ook gebruik worden gemaakt van ER-diagrammen. Ter verduidelijking bevindt zich hieronder een willekeurige uitwerking van de *logical* view, dat middels een klasse-diagram is gerealiseerd.



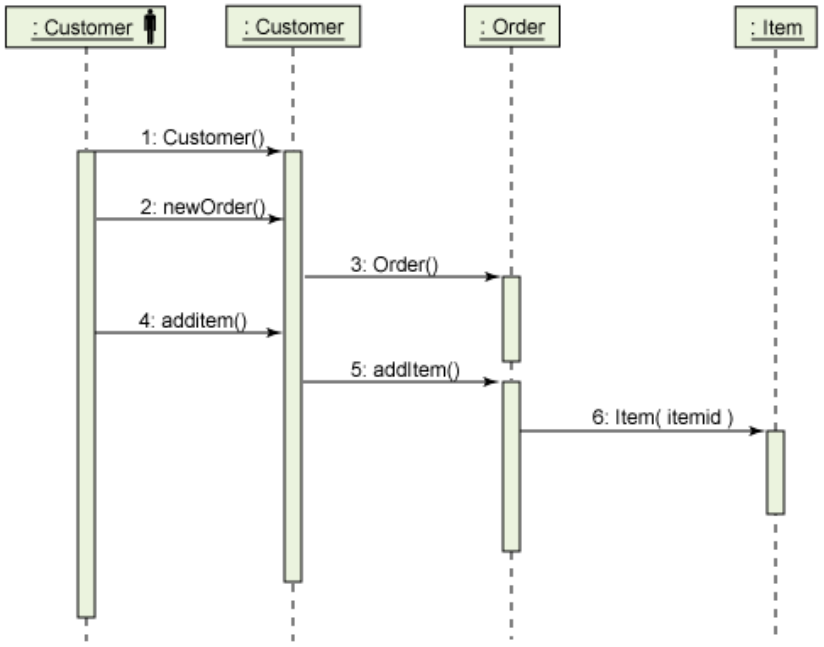
Figuur 11. Een view dat middels *klasse-diagram* is gepresenteerd [37]

Hoewel handig, is het gebruik van enkel een klasse-diagram vaak niet voldoende om een compleet beeld van de situatie te verkrijgen. Klasse-Diagrammen zijn namelijk statisch van karakter, hierdoor kan er geen goed inzicht in de dynamiek van een systeem verkregen worden. Zo kan men niet zien hoe een systeem op gebruikersinput zal reageren. Om alsnog inzicht te kunnen verkrijgen in de dynamiek van een systeem en de manier waarop objecten met elkaar interacteren, kan men ervoor kiezen om gebruik te maken van collboration- en sequentie diagrammen.

Hieronder staan een twee concrete *logical* view voorbeelden, waarbij de ene view doormiddel van een collaboration-diagram is opgezet en de andere view doormiddel van een sequentie-diagram.



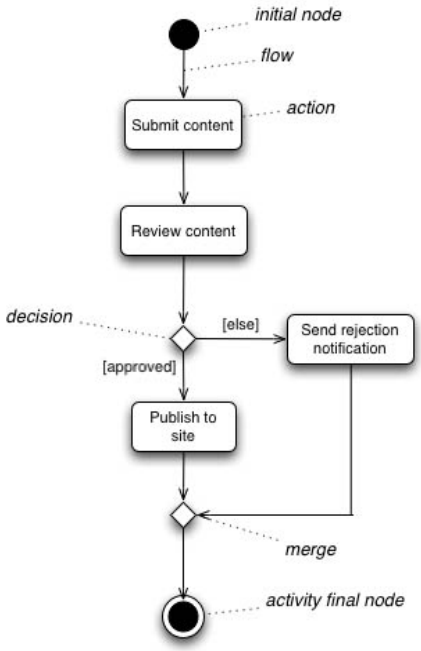
Figuur 12. Een view dat middels een *collaboration-diagram* is gepresenteerd[37]



Figuur 13. Een view dat middels een *sequentie diagram* is gepresenteerd[37]

2. Process view

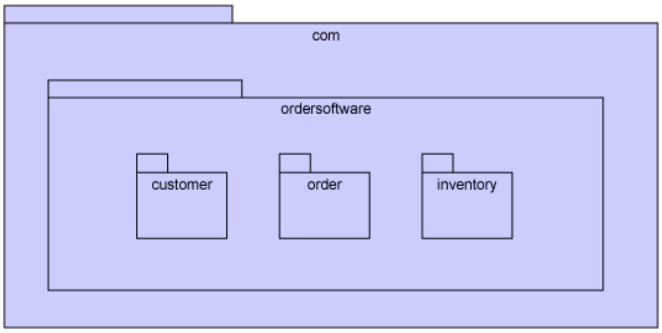
De process view beschrijft de processen binnen het systeem en geeft de manier weer waarop de onderdelen met elkaar communiceren. Het houdt zich voornamelijk bezig met non-functionele aspecten (zoals prestaties en beschikbaarheid) van een systeem. Met behulp van modelleertechnieken zoals activiteiten diagrammen kunnen de processen weergeven worden. Hieronder staat een voorbeelduitwerking van een willekeurig process view, dat doormiddel van een activiteiten diagram is gemaakt



Figuur 14: Een view dat middels een *activiteiten diagram* is gepresenteerd[38]

3. Development view

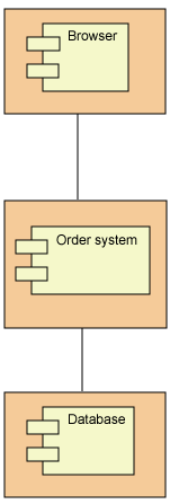
De development view laat zien uit welke modules en lagen een systeem bestaat en hoe deze zijn georganiseerd. Modules zijn nog grotere bouwsteden dan klassen en objecten. Ook kan deze view gebruikt worden om de omgeving te bepalen waarin het systeem ontwikkeld zal worden. Zo kan deze view als basis dienen voor het verzamelen van requirements, het verdelen van werk en taken aan ontwikkelaars, de kosten berekenen, de planning evalueren, de voortgang van het project en werkzaamheden in de gaten te houden en de herbruikbaarheid, transporteerbaarheid en beveiliging van softwaremodulen te bepalen. De development view vormt als het ware de basis voor het realiseren van een product. Om de modules in kaart te brengen kan men onder ander gebruik maken van package diagrammen. Hieronder staat een voorbeeld van een view dat middels een package diagram is gemaakt.



Figuur 15: Een view dat middels een *package diagram* wordt gepresenteerd[37]

4. Physical view

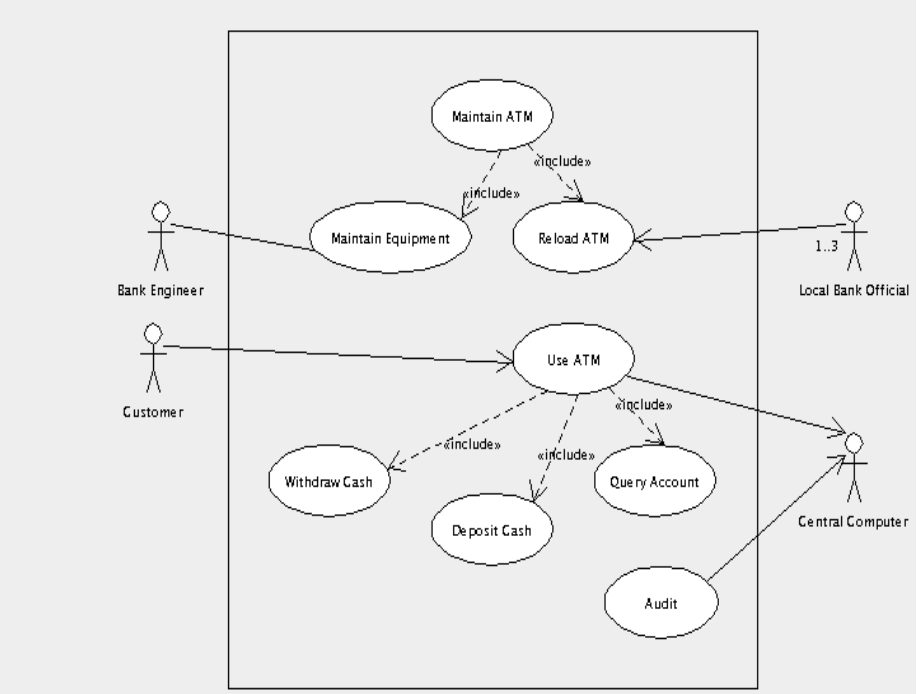
De physical view beschrijft hoe het systeem wordt geïnstalleerd, zal werken en welke hardware er gebruikt moet worden om de werking van het systeem te realiseren. Deze view houdt voornamelijk rekening met de non-functionele requirements (prestaties, beschikbaarheid, betrouwbaarheid en schaalbaarheid). Vaak wordt er gebruik gemaakt van deployment diagrammen om vanuit deze view de wijze waarom systemen wordt geïnstalleerd te modelleren.



Figuur 16: Een view dat middels een *deployment diagram* is gepresenteerd[37]

5. Scenario's

Soms wordt deze view ook gerefereerd als de use-case view. Aan de hand van deze view kunnen de architectuurbeschrijvingen van de vorige vier views doormiddel van scenario's worden gecontroleerd. Zo beschrijven de scenario's vanuit een gebruikersperspectief de interactie tussen objecten en processen. Deze scenario's worden meestal aan de hand van use-cases en use-case specifications beschreven. Hieronder staat een willekeurige voorbeeld van een use-case view.



Figuur 17: Een view dat middels een *use case* wordt gepresenteerd[39]

Bevindingen ten aanzien van het model

Het Kruchten 4+1 view model is een nuttige methode om systemen te analyseren en te documenteren. De kracht van dit model ligt vooral in het feit dat stakeholders naar diverse lagen van een systeem kunnen kijken. Zo is het voor software engineers handig om het systeem eerst vanuit de physical view bekijken en daarna vervolgens vanuit de proces view. Voor gebruikers, klanten en data-specialisten zou het juist interessant zijn om het systeem vanuit de logical view te bekijken. Projectmanagers kunnen hun systeemontwikkelingtraject analyseren door het systeem vanuit de development view te bekijken. Op deze manier kunnen diverse stakeholders eenvoudig naar aspecten zoeken die ze nodig hebben in het systeem.

Aan de andere kant heeft het Kruchten 4+1 view model het nadeel dat dit teveel neigt naar het beschrijven van low level structuren (operationeel niveau) in plaats van high level structuren (strategisch niveau) [54]. Dit model is heel applicatiegericht, waardoor het minder geschikt is om architecturen op een hoger niveau te beschrijven. Ook dient er in een bepaalde volgorde gewerkt te worden om views te maken, waardoor verschillende mensen niet altijd parallel kunnen werken aan de views.

7.2.2 het Zachman-raamwerk

De basis voor dit enterpriseraamwerk werd in 1987 door John Zachman gelegd. In z'n artikel "A framework for information systems architecture" [40] beschrijft hij een raamwerk om architecturen van informatiesystemen op te stellen en te beheren. Zijn idee was dat de architectuur voor informatiesystemen zou kunnen worden geïnspireerd door architecturen uit de wat oudere en meer volwassen disciplines. Zoals die uit de productie (voertuigen) en constructie (gebouwen) wereld. Hij vond dat beide disciplines veel overeenkomsten met elkaar vertoonden en dat ze samengevoegd konden worden om tot een nieuw algemeen model te komen.

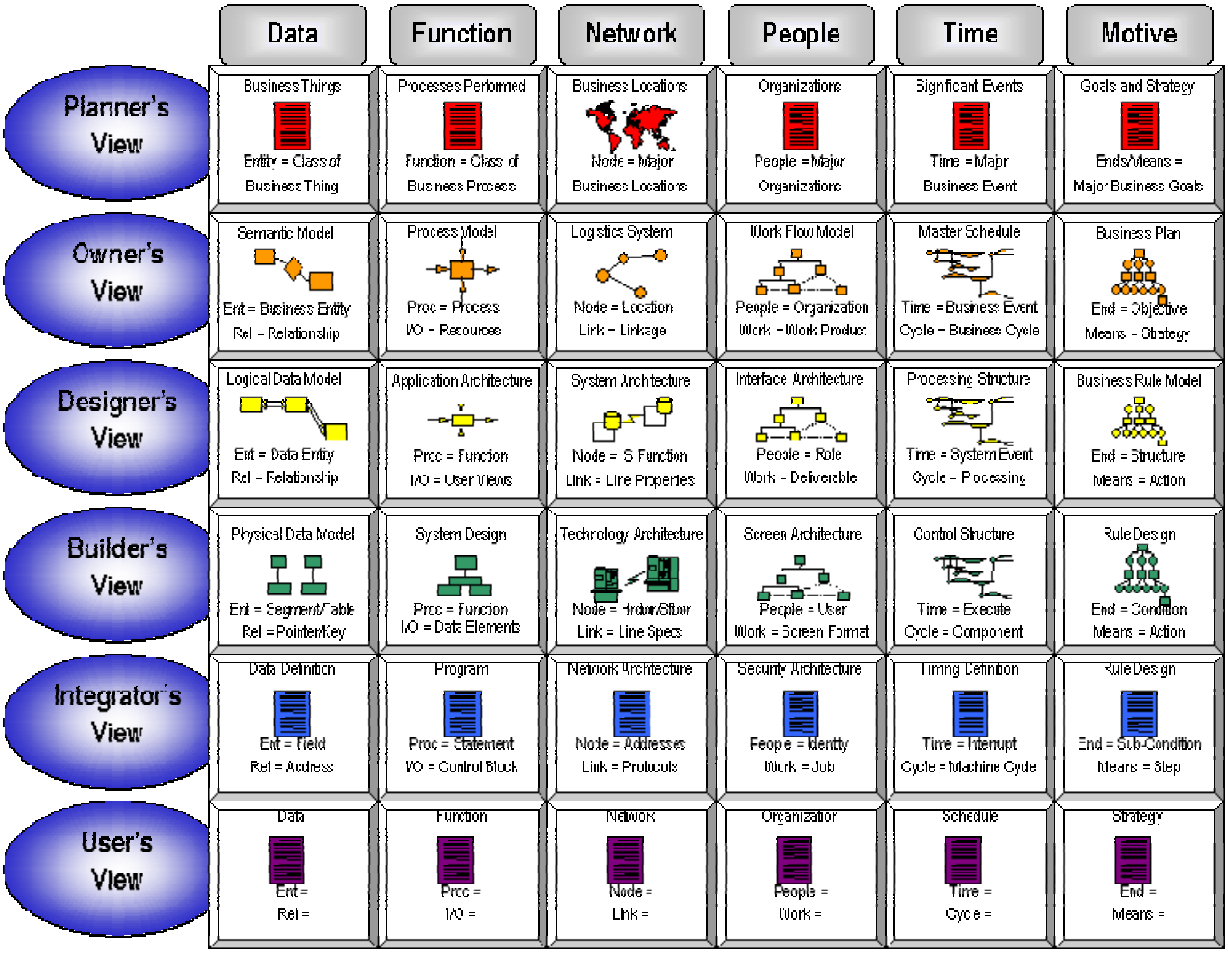
Aan de hand van het Zachman-raamwerk kunnen organisaties de informatie-infrastructuren van hun onderneming bekijken, analyseren of naar andere mensen toe communiceren. Het verstrekt als het ware een soort blauwdruk van zowel de huidige als toekomstige informatie-infrastructuur van een organisatie.

Het Zachman-raamwerk maakt onderscheid tussen twee dimensies in de vorm van betrokken partijen en aspecten (soort informatie) die van belang zijn voor het managen van het ontwikkelproces voor een informatiesysteem. De eerste dimensie wordt gevormd door de stakeholders die betrokken zijn bij de ontwikkeling van een informatiearchitectuur. Dit zijn de planner, de eigenaar, de ontwerper, de bouwer, de onderaannemer van het informatiesysteem en tenslotte de gebruiker. De tweede dimensie bestaat uit de elementaire vragen: wat, hoe, waar, wie, wanneer en waarom? In informatiesystemen-begrippen wordt dit vertaald naar gegevens, functies, netwerklocaties, personen, tijd en motivatie [41].

Het idee achter dit raamwerk is dat beide dimensies onafhankelijk van elkaar kunnen variëren. Zo kunnen er uiteindelijk 36 verschillende soorten perspectieven (viewpoints) gemaakt worden. Er zijn namelijk zes stakeholders en zes aspecten. Wanneer men deze met elkaar vermenigvuldigd komt men tot 36 mogelijke modellen.

Structuur

In het Zachman-raamwerk worden deze modellen middels een tabel weergegeven. Deze tabel beschikt over zes rijen en zes kolommen. Bij elkaar zijn dit dus (6*6) 36 unieke cellen of aspecten. De kolommen representeren de aspecten die van belang zijn voor het managen van het ontwikkelproces en de rijen staan voor de betrokken partijen. Op de volgende pagina staat een voorbeeld van een Zachman-raamwerk tabel.



Figuur 18: De verschillende viewpoints binnen het Zachman raamwerk [42]

Tabeleigenschappen

- Iedere kolom vertegenwoordigt een verschillende abstractie;
- Ieder model binnen een kolom is uniek;
- Iedere rij representeert een uniek perspectief;
- Iedere cel is uniek..

Betekenis van de rijen

Rij	Beschrijving
Planner view	Het systeem (of de architectuur) wordt hier vanuit het perspectief van de planner bekeken. Dit betekent dat de viewpoints die gerelateerd zijn aan deze view, enkel concerns adresseert die gerelateerd zijn tot de planning.
Owner view	Het systeem (of de architectuur) wordt hier vanuit het perspectief van de eigenaar bekeken. Dit betekent dat viewpoints die gerelateerd zijn aan deze view, enkel concerns adresseert die relevant zijn voor de eigenaar.
Designer view	Het systeem (of de architectuur) wordt hier vanuit het perspectief van de ontwerper bekeken. Dit betekent dat viewpoints die gerelateerd zijn aan deze view, enkel concerns adresseert die te maken hebben met het ontwerp van het systeem.

Builder view	Het systeem (of de architectuur) wordt hier vanuit het perspectief van de bouwer of ontwikkelaar bekeken. Dit betekent dat viewpoints die gerelateerd zijn aan deze view, enkel concerns adresseert die te maken hebben met de ontwikkeling van het systeem.
Integrator view	Het systeem (of de architectuur) wordt hier vanuit het perspectief van de onderaannemer bekeken. Dit betekent dat viewpoints die gerelateerd zijn aan deze view, enkel concerns adresseert van de onderaannemer.
User view	Het systeem (of de architectuur wordt hier vanuit het perspectief van de gebruiker bekeken. Dit betekent dat de viewpoints die gerelateerd zijn aan deze view, enkel concerns adresseert die van belang zijn voor de gebruiker van het systeem.

Betekenis van de kolommen

Kolom	Beschrijving
“Who?”/Wie (personen)	Dit zijn alle personen die een relatie hebben met betrekking tot het systeem of de architectuur.
“When?”/Wanneer (tijd)	Laat de tijd of de relatie tussen gebeurtenissen zien, dit is vooral handig wanneer men de planning maakt en processen ontwerpt.
“Why?”/Waarom (motivatie)	Hier wordt de reden van het systeem beschreven. Zo komen onder andere de missie, doelstellingen en het businessplan aan bod.
“What?” / Wat (gegevens)	In deze kolom worden de entiteiten beschreven die in elk perspectief (viewpoint) van de enterprise voorkomt. Voorbeelden van entiteiten zijn business objecten, systeemdata, relationele tabellen en tekstbox beschrijvingen.
“How?” / Hoe (functies)	Hier worden de functies beschreven die zich binnen ieder perspectief (viewpoint) bevindt.
“Where?” / Waar (locatie)	Dit zijn alle locaties binnen de enterprise. Voorbeelden zijn geografische locaties, netwerk locaties en geheugen locaties.

Typerende kenmerken

- Men gaat er vanuit dat een systeem volledig aan de hand van de vragen waarom, wie, wat, hoe, waar en wanneer gemodelleerd kan worden en dat zes perspectieven in principe voldoende zijn om alle modellen te ontwikkelen die nodig zijn om het systeem te kunnen ontwikkelen.

- Het Zachman-raamwerk is relatief eenvoudig qua opzet. Zo wordt er niet expliciet gebruik gemaakt van een ingewikkelde technische terminologie en is het daardoor goed te begrijpen voor zowel technici als non-technici.

- Het is een hulpmiddel om te kunnen plannen.

- Men kan domeinen vereenvoudigen zonder de complexiteit van het geheel te missen. Hierdoor is dit raamwerk uitstekend geschikt als hulpmiddel om problemen op te lossen.

Bevindingen ten aanzien van het model
Op basis van het Zachman-raamwerk is het voor IT-architecten en systeemontwikkelaars mogelijk om complexe, technologische of methodische keuzes te maken, die zowel door het algemeen- als het technisch-management als belangrijk worden ervaren.

Door de loop der jaren is dit raamwerk uitgegroeid tot de facto-standaard binnen de enterprise architectuurwereld. Zo hebben grote multinationals als Volkswagen, Boeing, General Motors en Bank of America om hun architecturen op te zetten gebruik gemaakt van dit raamwerk. John Zachman kan dan ook terecht worden beschouwd als één van de grondleggers van het architectuur denken. Veel van de huidige literatuur omtrent het werken met enterprise architecturen is gebaseerd op het artikel van Zachman. IEEE 1471 is hier een concreet voorbeeld van.

Er wordt in dit raamwerk niets gezegd over de wijze (dus het proces) waarop een architectuur tot stand kan of moet komen. Dit door mij als één van de grootste tekortkomingen van dit raamwerk beschouwd.

7.2.3 RM-ODP (ISO 10746)

Het ISO Reference Model of Open Distributed Processing (RM-ODP) [33, 41, 44] is een architectuurraamwerk dat zich vooral richt op de specificatie van gedistribueerde informatiesystemen. RM-ODP kent vijf basis viewpoints ten aanzien van een systeem en zijn omgeving. Deze zijn het enterprise, information, computational, engineering en technology viewpoint.

- Enterprise viewpoint
  De enterprise viewpoint representeert het business model en de business requirements. Dit viewpoint wordt voornamelijk gebruikt om de zakelijke behoeften van een organisatie aan het systeem te koppelen. Daarom is het belangrijk dat de views die aan de hand van deze viewpoint worden gemaakt, door alle stakeholders in de business environment begrepen worden. De belangrijkste concerns van dit viewpoint zijn:

  - De doelstelling, scope en richtlijnen voor het systeem;
  - Rollen die door het systeem vervuld worden;
  - Activiteiten die door het systeem worden uitgevoerd;
  - Beleidsreglementen ten aanzien van het systeem.

  In de enterprise viewpoint wordt een systeem en zijn environment weergeven als een gemeenschap van objecten. Deze objecten worden gedefinieerd in termen van:

  - Enterprise objecten die de gemeenschap vormen;
  - Rollen die door ieder object vervuld worden;
  - Regelgeving of beleid tussen de verschillende objecten en hun rollen;

- Beleid ten aanzien van het creëren, gebruiken en verwijderen van resources bij objecten;
- Beleid ten aanzien van het configureren van objecten en het geven van rollen aan objecten;
- Beleid ten aanzien van de environment (gebruikers) die het systeem bestuurt, toestemmingen, verplichtingen en gedrag.

- Information viewpoint
  De information viewpoint is voornamelijk gericht op de semantiek van informatie en de manier waarop informatie verwerkt wordt.

- Computational viewpoint
  De computational viewpoint is erop gericht de functionele decompositie van het systeem in termen van objecten die een interactie hebben met de interface van het systeem te beschrijven. In andere woorden dit viewpoint beschrijft de functionaliteiten die door het systeem worden aangeboden en laat zien hoe deze in elkaar zitten.

- Engineering viewpoint
  De engineering viewpoint focust zich op de mechanismen en functies tussen de diverse componenten in een systeem. Zo beschrijft het op welke manier het systeem gegevens verwerkt en levert het op deze manier de benodigde informatie aan, die het mogelijk maakt om de gewenste functionaliteiten te ontwikkelen en te implementeren in het systeem.

- Technology viewpoint
  De technology viewpoint concentreert zich op de technische aspecten van het systeem. Zo beschrijft het de keuze van de techniek die in het systeem gebruikt zal worden, hoe de specificaties worden geïmplementeerd, worden er relevante technologieën gespecificeerd en functioneert dit viewpoint als een ondersteuning voor het testproces in het algemeen.

Bevindingen ten aanzien van het model
Het grote nadeel van RM-ODP is dat het voornamelijk gericht is op de communicatie tussen ontwikkelaars van verschillende systemen en niet tussen stakeholders van hetzelfde systeem [53].

7.2.4 Problematiek ten aanzien van de view modellen

Wanneer de drie view modellen uit de vorige paragrafen met elkaar vergeleken worden, komen we al gauw tot de conclusie dat deze modellen onderling flink van elkaar verschillen. Dit is een herkenbaar fenomeen bij veel view modellen (of architectuur raamwerken). Er bestaan namelijk tientallen verschillende view modellen die ieder een eigen methodiek hebben. Hoe meer view modellen er zijn, des te moeilijker het voor systeemontwikkelaars of IT-architecten wordt om een keuze te maken.

Op de volgende pagina bevindt zich een overzicht van de meest gangbare view modellen van dit moment. Overigens dient men er rekening mee te houden dat dit maar een

beperkte opsomming is en dat er ongetwijfeld meer view modellen zullen bestaan dan er hieronder zullen worden aangegeven. Het inhoudelijk analyseren en vergelijken van view modellen valt buiten de scope van dit onderzoek. Echter om een goed beeld te kunnen verkrijgen omtrent de problematiek rondom het maken van een viewpointselectie, is het noodzakelijk om een overzicht van de diverse view modellen te presenteren .

Overzicht van overige architectuur raamwerken of view modellen

View model	Viewpoint(s)
2+2 model	Context, Technical Infrastructure, Conceptual, Development
4+1 model	Logical, Process, Development, Physical, Scenarios
ARIS	Organizational, Data, Control, Function, Product/Service
Boar	Infrastructure, Data, Applications, Organization
DYA	Business (Product, Proces, Organisatie), Informatie (Gegevens, Applicatie), Technisch (Middleware, Platform, Netwerk)
Herzum/Sims	Functional, Application, Technical, Project Management
IAF	Business, Information, Information Systems, Technology Infrastructure, Contextual, Conceptual, Logical, Physical, Transformational, Business and ICT System, Security, Governance
ISA	Data, Function, Network, People, Time, Motivation, Contextual, Conceptual, Logical, Physical, Implementation, Out-of-Context
IFW	Organization (Strategy, Structure, Skills), Business (Data, Function, Workflow, Solution), Technical (Interface, Network, Platform)
MAD	Inter-organizational, Organizational, Process, Information, Application, Distribution, Configuration
Maier/Rechtin	Data, Behavior, Form, Purpose, Performance, Managerial
MArch	Product, Proces, Organisatie, Informatievoorziening, Infrastructuur
RM-ODP	Enterprise, Informational, Computational, Engineering, Technology
Siemens	Conceptual, Module, Execution, Code
Tapscot	Business, Work, Information, Application, Technology
TOGAF	Business (People, Process, Function, Business Information, Usability, Performance), Engineering (Security, Software Engineering, Data, System Engineering, Communications Engineering), Operations (Security, Software, Data, Computing/Hardware, Communications), Acquirers (Building Blocks Cost, Standards)

Tabel is overgenomen uit [34]

Terminologie

Niet alleen bestaan er veel verschillende view modellen (of architectuurraamwerken), maar ook de gebruikte terminologie kan zelfs per view model verschillen. Zo kunnen verschillende termen voor dezelfde aspecten gebruikt worden of kunnen dezelfde termen per view model een andere betekenis hebben.

Wanneer meerdere ontwikkelaars meerdere architecturen of systemen aan de hand van verschillende view modellen ontwikkelen, zal het verschil in terminologie uiteindelijk ertoe leiden dat de communicatie tussen deze ontwikkelaars enigszins bemoeilijkt wordt.

Het verschil in terminologie komt duidelijker naar voren wanneer men enkele view modellen met elkaar vergelijkt. Wat bijvoorbeeld in IFW “Strategy” heet, valt bij Boar onder “Organisation”. Wat in IFW “Data” heet valt in IAF uiteen in “Information” en “Information Systems’. Hieronder volgt een kort overzicht van de verschillen en overeenkomsten tussen een aantal termen, dat door diverse view modellen wordt gehanteerd.

Overzicht gebruikte terminologie in architectuurraamwerken of view modellen

IFW	Boar	DYA	IAF	MAD	Zachman
Strategy		Organisation	Business	Organisational	Motivation
Structure		Organisation	Business	Organisational	Network
Skills		Organisation	Business	Organisational	
Data	Data	Gegevens	Information Information Systems	Information	Data
Function	Application	Proces	Information Information Systems	Process	Function
Workflow	Application	Proces	Business	Process	Time
Solution		Product	Business	Interorganisational	
Interface		Applicatie	Information Systems	Application	Network Function
Network	Infrastructure	Netwerk	Technology Infrastructure	Configuration	Network
Platform	Infrastructure	Platform Middleware	Technology Infrastructure	Configuration Distribution	Network

Tabel is overgenomen uit [34]

Zoals men ziet hoeft het ruime aanbod aan view modellen niet per definitie het werken met viewpoints te bevorderen. Er zijn namelijk veel verschillende opties mogelijk en hoe meer keuze er is, des te moeilijker het wordt om een keuze te maken. Dit is een typisch voorbeeld van wat ik aan het begin van dit hoofdstuk bedoelde met door de bomen het bos haast niet meer zien. Het vergt daarom enige expertise en ervaring om de juiste keuze te kunnen maken.

Mede veroorzaakt door de rijkheid en grote diversiteit aan view modellen is het soms moeilijk om aan mensen uit te leggen wat een viewpoint precies is en wat men ermee kan doen.

7.2.4 Classificatie raamwerk voor viewpoints

In de vorige paragraaf heeft men een voorproefje kunnen krijgen van de hoeveelheid view modellen (of architectuurraamwerken) die momenteel in de omloop zijn. Het moge inmiddels duidelijk zijn, dat er bij ieder view model ook viewpoints horen. Er bestaan vele tientallen wellicht honderden verschillende viewpoints, hierdoor is het praktisch onmogelijk om al deze viewpoints uit het hoofd te kennen en een overzicht te houden.

Om toch enige order in deze chaos van opties te creëren, kan er gebruik worden gemaakt van een viewpoint-classificatieraamwerk.

Soorten viewpoints
Onderzoekers van het telematica instituut [29] in Enschede hebben geprobeerd de verschillende viewpoints te categoriseren en een classificatieraamwerk te creëren. Dit raamwerk maakt het voor IT-architecten of andere personen die met viewpoints dienen te werken eenvoudiger om viewpoints te kiezen.

Over het algemeen kan er een onderscheid worden gemaakt tussen drie verschillende soorten viewpoints, namelijk ontwerpondersteunende, beslissingondersteunende en informerende viewpoints.

- Ontwerpondersteunende viewpoints:
  Deze viewpoints zijn er voor bedoeld om IT-architecten, systeemontwerpers of andere betrokkenen die te maken hebben met het ontwerp van een architectuur, ondersteuning te bieden bij het ontwerpproces.

- Beslissingsondersteunende viewpoints:
  Deze viewpoints biedt directeuren, managers en overig leidinggevend personeel, ondersteuning bij het besluitvormingsproces.

- Informerende viewpoints:
  Aan de hand van deze viewpoints kunnen stakeholders over een architectuur geïnformeerd worden. om zo meer begrip te krijgen, hogere betrokkenheid te creëren en mensen te kunnen overtuigen van bepaalde keuzes die zijn gemaakt.

Soort Viewpoint	Stakeholder(s)	Doelstelling	Voorbeelden
Ontwerp	architect, softwareontwerper, business proces ontwerper	Navigeren, ontwerpen, ontwerpbeslissingen ondersteunen en alternatieven vergelijken.	UML diagram, Testbed diagram
Beslissend	manager, CIO, CEO	Beslissingen maken.	cross-reference table, landscape map, lijst, management-rapportage
Informerend	werknemer, klant	Uitleggen, overtuigen, en steun verkrijgen.	Animatie, tekenfilm, proces illustratie, kaart

Om een beter beeld te kunnen verkrijgen van de diverse viewpoint categorieën, zal er op de volgende pagina's per hoofdcategorie een aantal viewpoints op basis van willekeurigheid worden behandeld. Deze voorbeelden zijn gebaseerd op [29]. Overigens dient er wel vermeld te worden dat hoewel de bijbehorende viewpoint-specificaties voor een groot deel gebaseerd zijn op de viewpoint-specificatie zoals die door IEEE 1471 is

geïntroduceerd, wijken ze enigszins toch af van de viewpoint-specificatie sjabloon dat in IEEE 1471 gehanteerd wordt. Zo wordt er in plaats van vijf viewpoint componenten gebruikt gemaakt van een zestal componenten.

Ontwerpondersteunende viewpoints

Ontwerpondersteunende viewpoints [29] worden in vier subcategorieën verdeeld. Hierbij wordt onderscheid gemaakt tussen compositie-, samenwerking-, support- en realisatie viewpoints. Deze hoofdcategorie kent een groot hoeveelheid viewpoints, deze viewpoints worden in vier subcategorieën verdeeld.

Subcategorie	Viewpoint(s)
Compositie	Organisatie, Business-functie, Business-proces, informatiestructuur, applicatiestructuur, applicatiegedrag en infrastructuur viewpoint.
Samenwerking	Tussen actoren, business processen en applicaties.
Support	Product, applicatie- en infrastructuurgebruik
Realisatie	Service realisatie, implementatie en deployment

- Organisatie viewpoint

  Dit viewpoint laat de interne structuur van een organisatie (bedrijf, afdeling, vereniging, etc.) zien. Aan de hand van dit viewpoint kan men bijvoorbeeld een organisatie (leiding en verantwoordelijkheden) beter begrijpen.

Attribuut	Beschrijving
Viewpoint naam:	Organisatie viewpoint
Stakeholders:	Enterprise, processen, domeinexperts architect, managers, werknemers
Concerns:	Organisatiestructuur, competenties, verantwoordelijkheden, autoriteit
Doelstelling:	Ontwerpen, beslissen, informeren
Abstractie niveau:	Detail niveau
Lagen:	Business Laag
Aspecten:	Actief

  Views die gerelateerd zijn aan dit viewpoint kunnen bijvoorbeeld middels een organogram weergegeven worden.

- Business functie viewpoint

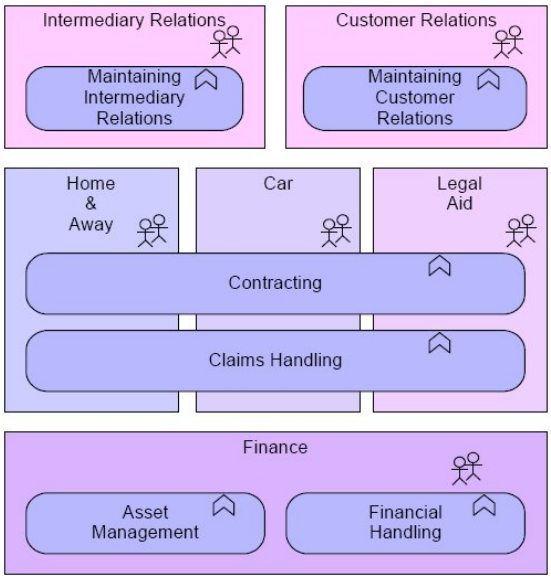
  Dit viewpoint laat de belangrijkste (zakelijke) processen samen met de bijbehorende informatiestromen en producten zien die zich binnen een organisatie afspelen.

  Business functies zijn bedoelt om te analyseren welke primaire processen (ongeacht organisatorische veranderingen of technologische ontwikkelingen) het meest stabiel zijn in een organisatie. De business functie viewpoint levert daarnaast ook strategisch inzicht in de processen die zich binnen een organisatie afspelen.

Aan de hand van dit viewpoint kunnen bijvoord de zwakke punten van een organisatie worden ontdekt of kunnen processen hergestructureerd worden.

Attribuut	Beschrijving
Viewpoint naam:	Business functie viewpoint
Stakeholders:	Enterprise, processen, domeinarchitect
Concerns:	Organisatiestructuur, competenties, verantwoordelijkheden, autoriteit
Doelstelling:	Ontwerpen
Abstractie niveau	Coherentie niveau
Lagen:	Business Laag
Aspecten:	Gedrag (actief)

Views die gerelateerd zijn aan dit viewpoint kunnen bijvoorbeeld aan de hand van de moduleertaal Archimate of doormiddel van ORM diagrammen gerepresenteerd worden.



Figuur 19: Business functies en hun afdelingen in Archimate [29]

- Applicatiestructuur viewpoint

Dit viewpoint laat de structuur van één of meerdere applicaties of componenten zien. Een applicatiestructuur viewpoint is voornamelijk handig bij het ontwerpen of begrijpen van applicaties of componentstructuren en hun data. Dit wordt vaak gezien als een eerste stap in het bouwen van een systeem (architectuur) of het migreren van meerdere systemen.

Component / Slot	Beschrijving
Viewpoint naam:	applicatiestructuur viewpoint
Stakeholders:	Enterprise, applicaties, domeinarchitect
Concerns:	Applicatiestructuur, consistentie en compleetheid, verminderen van complexiteit

Doelstelling:	Ontwerpen
Abstractie niveau:	Detail niveau
Lagen:	Applicatie Laag
Aspecten	Gedrag (actief)

Views die gerelateerd zijn aan dit viewpoint kunnen bijvoorbeeld aan de hand van functionele decompositie diagrammen gerepresenteerd worden.

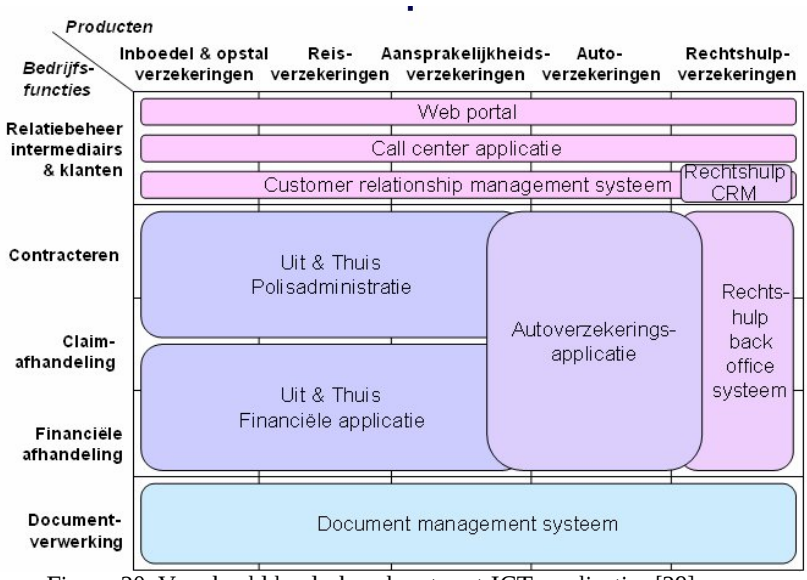
Beslissingsondersteunende viewpoints

Beslissingsondersteunende viewpoints [29] helpen managers of andere beslissingsmakers (doormiddel van projecties, analyse technieken en modellen) bij het maken van beslissingen en bieden meer inzicht in domein relaties. Beslissingsondersteunende views kunnen verschillende vormen aannemen. Een aantal voorbeelden hiervan zijn overzichtslijsten, datamatrices, managementrapportages en landschapkaarten.

- Landschapskaart viewpoint

  De kracht van deze viewpoint ligt vooral in de leesbaarheid ervan. Zo kunnen mensen die niet specifiek geschoold zijn in het lezen van complexe modellen, de gepresenteerde informatie begrijpen en aan de hand van deze informatie kunnen vervolgens besluiten worden genomen.

Attribuut	Beschrijving
Viewpoint naam:	landschapkaart viewpoint
Stakeholders:	IT-architect , beslissingsmakers zoals IT-managers, CEO's, CIO's
Concerns:	Leesbaarheid, vermindering van complexiteit, vergelijken van alternatieven
Doelstelling:	Besluitvorming
Abstractie niveau:	Overzichtsniveau
Lagen:	Business- en applicatielaag
Aspecten:	Actief



Figuur 20: Voorbeeld landschapskaart met ICT-applicaties [29]

Informerende viewpoints

Informerende viewpoints zijn bedoeld om stakeholders te kunnen informeren en op de hoogte te brengen over aspecten van een systeem of architectuur.

- Procesillustratie viewpoint
  Dit viewpoint kan door IT-architecten of managers gebruikt worden om stakeholders te overtuigen en ze op de hoogte te brengen van ontwerpbeslissingen.

Attribuut	Beschrijving
Viewpoint naam:	Procesillustratie viewpoint
Stakeholders:	IT-architect , managers
Concerns:	Ontwerpbeslissingen zichtbaar maken, stakeholders overtuigen
Doelstelling:	Communicatie
Abstractie niveau:	Coherentie niveau
Lagen:	Business, applicatie, technologisch lag
Aspecten:	Actief, gedrag en passief

Basisregels voor viewpoints

Volgens [29] bestaan er over het algemeen vier primaire basisregels om viewpoints te kunnen maken en te manipuleren. Dit zijn selectie-, presentatie-, interactie- en interpretatieregels.

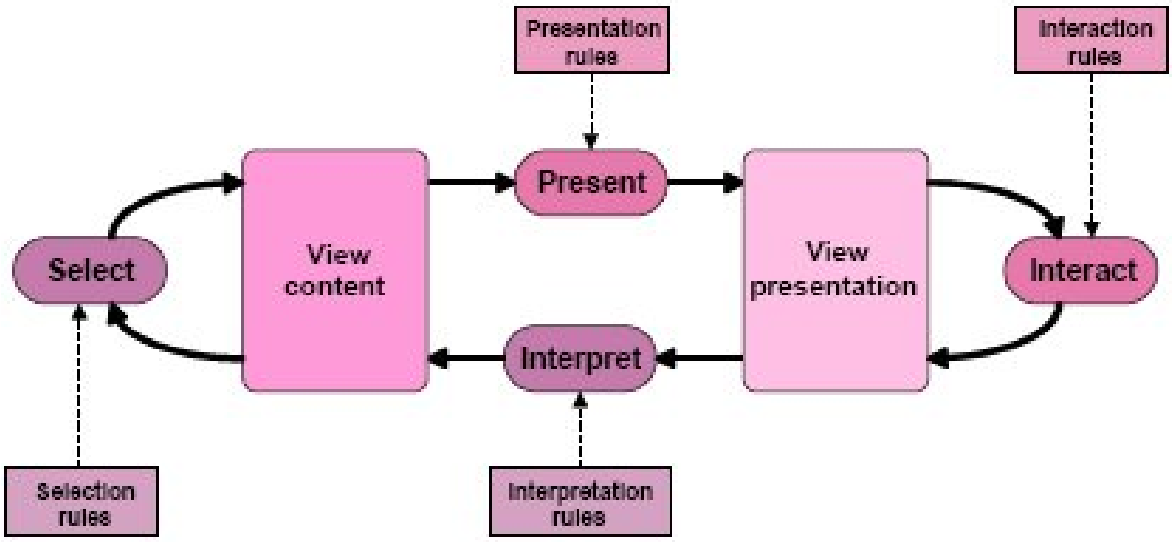
- Selectieregels
  Aan de hand van selectieregels wordt bepaald welke elementen van een architectuur (of systeem) weergeven moeten worden in een view. Deze regels bevatten niet alleen richtlijnen hoe elementen gekozen dienen te worden, maar ook regels om te kunnen abstraheren.

- Presentatieregels
  Aan de hand van presentatieregels wordt er bepaald hoe (kleur, lettertypes, symbolen, etc.) en op wat voor manier de gekozen elementen in een view gerepresenteerd dienen te worden.

- Interactieregels
  Aan de hand van interactieregels kunnen views die aan de hand van een viewpoint worden opgesteld gemanipuleerd worden.

- Interpretatieregels
  Deze interpretatieregels bepalen welke gevolgen de manipulaties die door de gebruiker op de views worden verricht, hebben op de onderliggende architectuur.

In het begin van dit onderzoek had ik aangegeven, dat een viewpoint als het ware een soort handleiding is om views te kunnen modelleren. Tot op heden was de invulling van wat deze handleiding dient te bevatten, nogal aan de vage kant. Zo biedt IEEE 1471 ook geen concrete informatie hierover. Deze regels bieden wat mij betreft een uitstekend beeld van wat er in deze handleiding beschreven dient te worden.



Figuur 21: Interactie van de basisregels op viewpoints en views [29]

Relatie van dit classificatieraamwerk ten aanzien van bestaande view modellen

Niet alle view modellen uit de vorige paragrafen kunnen helaas met dit classificatieraamwerk gecategoriseerd worden. Daar zijn de meeste modellen gewoon te divers voor. Een view model hanteert namelijk meerdere viewpoints, waardoor het onmogelijk is om een view model alleen in één categorie te plaatsen.

Maar wanneer de view modellen worden opgesplitst kunnen de individuele viewpoints wel degelijk binnen dit classificatieraamwerk gecategoriseerd worden. Zo kan het logical viewpoint van het Kruchten 4+1 view model onder ontwerpondersteunende viewpoints gecategoriseerd worden en kan het enterprise viewpoint van het RM-ODP view model onder de categorie beslissingsondersteunende viewpoints worden geplaatst. Op deze manier kunnen de viewpoints die zich in het view model overzichtstabel uit de vorige paragraaf bevinden alsnog aan de hand van dit classificatieraamwerk geclassificeerd worden.

Het hebben van een classificatieraamwerk creëert meer inzicht in het werken met viewpoints en views. Door viewpoints te kunnen classificeren krijgt men het overzicht weer terug. Daarnaast biedt een classificatieraamwerk de mogelijkheid om zelf nieuwe view modellen of viewpoints te ontwikkelen.

7.3 Hoe ontdekt men inconsistenties tussen views en hoe gaat men ermee om?

Een zeer handige mogelijkheid van het werken met viewpoints is dat men views met elkaar kan vergelijken. Op deze manier kunnen de modellen die aan de hand van viewpoints gecreëerd worden op fouten gecontroleerd worden. Het ontdekken en corrigeren van fouten leidt tot betere modellen, wat uiteindelijk ten goede komt aan de kwaliteit van het te ontwikkelen systeem.

Het proces om views en viewpoints op fouten te controleren wordt ook wel het ontdekken van inconsistenties genoemd.

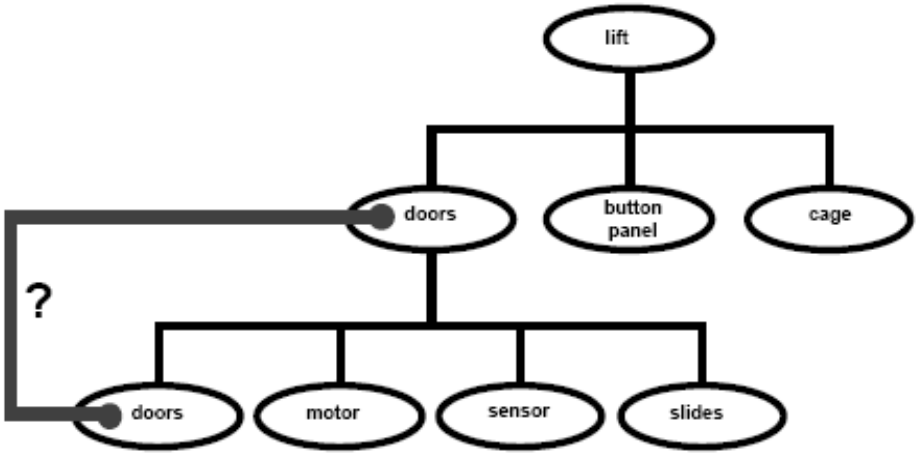
7.3.1 Wat zijn inconsistenties?

Bij het vastleggen van de verschillende views kunnen helaas inconsistenties ontstaan, dezelfde informatie kan immers zij het anders verwoord op verschillende plaatsen terug komen. Een goede consistentie tussen viewpoint-specificaties vormt dan ook een zeer belangrijk randvoorwaarde om producten op te kunnen leveren die aan klantverwachtingen voldoen. Om het begrip (in)consistenties te verduidelijken dienen de volgende sub-deelvragen beantwoord te worden:

- Hoe bepaal je of views consistent zijn?
- Hoe behoud je de consistentie tijdens het systeemontwikkelingsproces?
- Hoe ga je met inconsistenties om?

7.3.2 Hoe bepaal je of views consistent zijn?

Aan de hand van in-viewpoint checks en inter-viewpoint checks kan de consistentie tussen views en viewpoints gecontroleerd worden. In-viewpoint checks controleren de consistentie van de specificatie binnen dezelfde view en inter-viewpoint checks controleren de consistentie met die van andere viewpoints



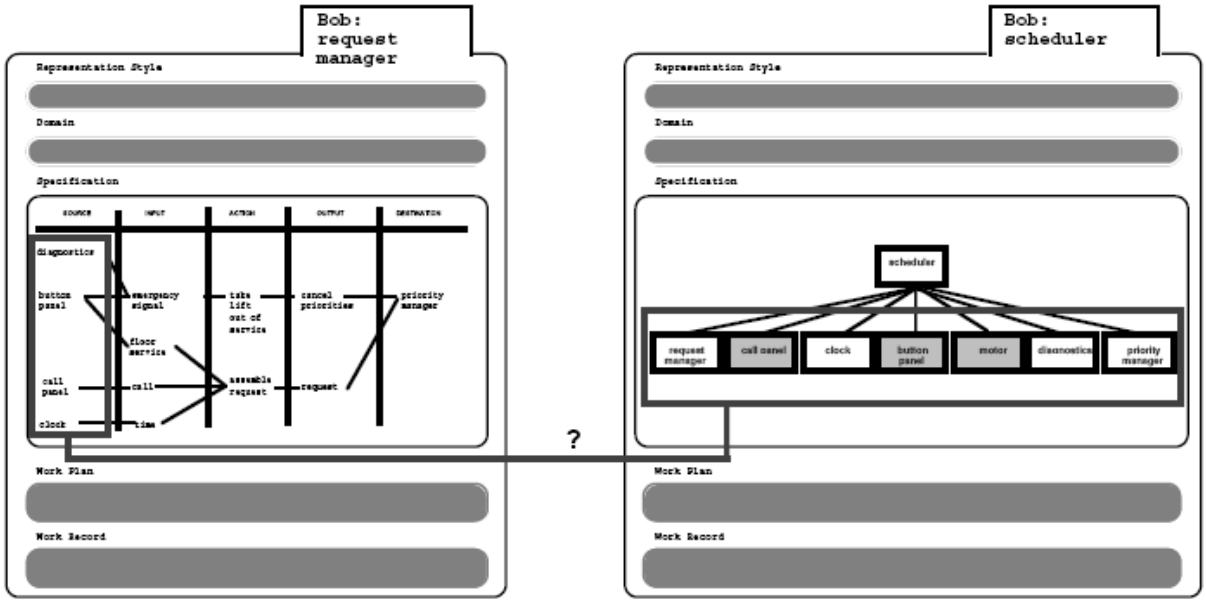
Figuur 22: Voorbeeld van een in-viewpoint check [28]

In het bovenstaande voorbeeld is duidelijk te zien dat er binnen deze view een consistentiefout aanwezig is. Zo kan het onderdeel “deur” geen onderdeel van zichzelf zijn. Doormiddel van een in-viewpoint check kunnen dit soort fouten vroegtijdig ontdekt worden en kunnen resources bespaard worden. Nu is dit een simpel voorbeeld, maar men moet zich voorstellen dat een consistentie check bij complexe en grote modellen uiteraard een veel complexer klus zal zijn.

Belemmerende factoren

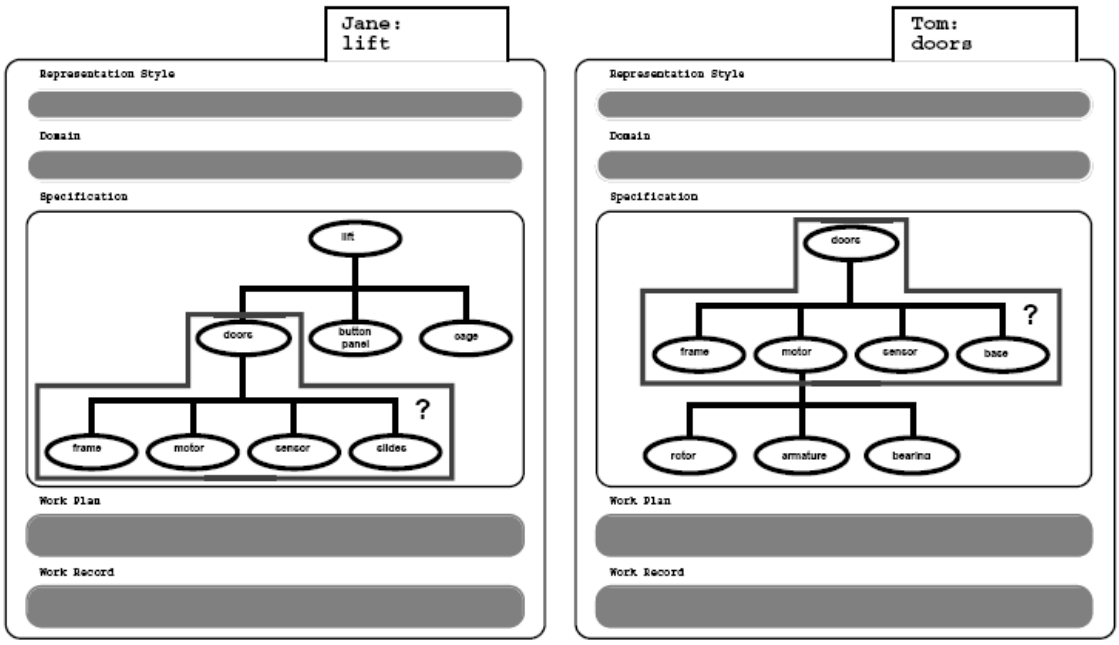
Het vergelijken van views die dezelfde representatiestijlen bevatten is nog goed te doen, maar het kan ook gebeuren dat er per view een andere representatiestijl wordt gebruikt. In zulke gevallen zal het achterhalen van inconsistenties aanzienlijk complexer worden. Per view en per representatiestijl dient er dan namelijk telkens weer een manier gevonden te worden om de representatiestijl van de ene view met de andere view te vergelijken. Aangezien er geen er momenteel geen standaard methodiek aanwezig is die deze handelingen beschrijven, is men hiervoor dan ook volledig afhankelijk van de vaardigheden en ervaring van de modelleur.

Hieronder is een voorbeeld te vinden van een situatie waarbij twee verschillende viewpoints (die tevens ook twee verschillende representatiestijlen bevatten) met elkaar vergeleken dienen te worden om ze zo op inconsistenties te kunnen controleren. Als de representatiestijlen fundamenteel van elkaar sterk verschillen kan het heel moeilijk en complex zijn om de modellen op fouten te controleren.



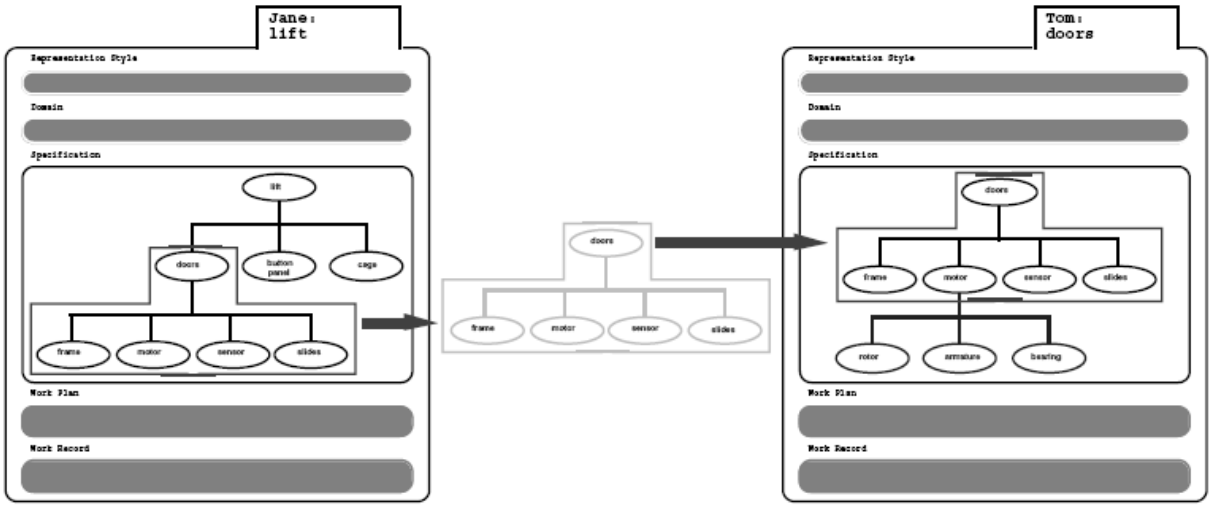
Figuur 23: Vergelijken van verschillende representatietechnieken [28]

Soms kunnen meerdere personen die dezelfde representatiestijlen hanteren verschillende ontwerpen voor hetzelfde onderdeel leveren. Zo kan het gebeuren dat de consistentie binnen een individuele viewpoint in orde is, maar wanneer verschillende viewpoints met elkaar vergeleken worden ze niet blijken consistent te zijn. In het voorbeeld op de volgende pagina zijn twee verschillende ontwerpen gemaakt voor de deur. Beide ontwerpen zijn consistent met elkaar en verschillen alleen qua structuur van elkaar. Het probleem is dat beide ontwerpen eigenlijk goed zijn, maar er kan uiteindelijk maar één ontwerp voor een onderdeel gekozen worden. In een laat stadia kunnen dit soort situaties een zwaar belemmerende werking op de projectvoortgang hebben. Het gebruik van viewpoints maakt het mogelijk dit soort foute vroeg te ontdekken.



Figuur 24: Voorbeeld van een inter-viewpoint check [28]

In veel gevallen kan men inconsistenties repareren door stukken uit een model letterlijk eruit te knippen en te verplaatsen naar een ander viewpoint.



Figuur 25: Voorbeeld van een inter-viewpoint tranfer [28]

7.3.3 Hoe ga je met inconsistenties om?

Zodra inconsistenties opduiken dienen de afwijkende modellen met elkaar vergeleken te worden. Men dient vervolgens te achterhalen wat er mankeert aan deze modellen en welke model correct is. De bovenstaande consistentie checks komen hierbij goed van pas.

Om modellen weer consistent te maken kan men in overleg besluiten om het deel van het model waar het verschil in zit van het ene viewpoint naar het andere over te plaatsen. Op deze manier wordt de consistentie opgelost.

Technieken om consistenties tussen views te kunnen achterhalen zijn de formele methodes LOTOS [45] en Z [46].

7.3.4 Hoe behoud je de consistentie tijdens het systeemontwikkelingsproces?

Bij kleine projecten waar er relatief eenvoudige systemen worden ontwikkeld, zal het minder moeilijk zijn om de consistentie gedurende een project te bewaren, dan wanneer men het over grote projecten en complexe systemen heeft.

Communicatie tussen stakeholders is van essentieel belang voor het behouden van de consistentie binnen het systeemontwikkelingproces. Zo dienen ontwikkelaars regelmatig zowel onderling als met de toekomstige gebruikers te overleggen. Daarnaast dienen er concrete afspraken en procedures worden opgesteld om later onduidelijkheden te voorkomen. Geen enkel maatregelen kan echter 100% garanderen dat de consistentie gewaarborgd blijft, maar zullen de systeemontwikkelaars wel helpen het risico te verminderen of dan wel te beheersen.

7.4 Hoe betrouwbaar is het werken met viewpoints?

Voor ontwikkelaars is het belangrijk om te weten of een systeemontwikkelingmethode betrouwbaar is. Het is echter heel moeilijk om op een concrete wijze de betrouwbaarheid van een methodiek in cijfers te weergeven. Om toch enige inzicht te kunnen geven over de betrouwbaarheid van deze methode zal ik hieronder de zwakheden van het werken met viewpoints proberen boven water te krijgen, te analyseren en vervolgens te vergelijken met traditionele systeemontwikkelingmethoden.

7.4.1 Kunnen views door externe of interne factoren beïnvloed worden?

Viewpoints zijn statisch van karakter en lijken me daardoor niet gevoelig voor externe of interne factoren. Views daarentegen zijn dynamisch en kunnen in tegenstelling tot viewpoints wel beïnvloed worden. Dit komt doordat views aan de hand van stakeholder concerns gegenereerd worden. Omgevingsfactoren kunnen de concerns van een persoon beïnvloeden en als concerns veranderen is het ook zeer waarschijnlijk dat een view mee veranderd.

De spelregels voor het maken van een view worden door de bijbehorende viewpoint-specificatie bepaald. Het is natuurlijk vanzelfsprekend dat wanneer een viewpoint-specificatie wordt aangepast de gerelateerde view ook automatisch mee veranderd.

Ook sociale factoren kunnen een view beïnvloeden. Dit heeft te maken met onze conceptie van de realiteit, deze is eerder subjectief dan objectief. Iedere persoon heeft namelijk een unieke conceptie op de wereld, waardoor de realiteit per persoon anders waar kan worden genomen. Aspecten zoals belangen, kennis en persoonlijk ervaringen kunnen dus views beïnvloeden.

7.4.2 Wat zijn de zwakheden van het werken met viewpoints?

- Views zijn gevoelig voor de concerns van stakeholders, waardoor domeinen niet objectief gemodelleerd worden;
- Hoe meer viewpoints er zijn des te complexer het wordt om views te managen en de consistentie tussen diverse modellen te behouden.
- Het werken met viewpoints brengt in vergelijking met traditionele systeemontwikkelingmethoden extra taken en procedures met zich mee. Zo moet

er meer gedocumenteerd worden en dienen views op consistentie gecontroleerd te worden.

7.4.3 Wat zijn de sterke punten van het werken viewpoints?

- De consistentiecontroles tussen de views zullen ongetwijfeld de kwaliteit van de modellen verbeteren, wat op den duur leidt naar een beter begrip van het te onderzoeken domein.

- Door expliciet vanuit het perspectief van een bepaald stakeholder naar het domein te kijken, lijkt het me waarschijnlijk dat er niet alleen meer, maar ook kwalitatief betere requirements ontdekt zullen worden. Dit zal op den duur tot een completer requirements engineering leiden.

7.4.4 In hoeverre is het werken met viewpoints betrouwbaarder dan traditionele systeem ontwikkelingmethoden?

Het werken met viewpoints kan beschouwd worden als een soort aanvulling op de huidige “traditionele” systeemontwikkelingmethodes. Alle handelingen die in een traditioneel systeemontwikkelingsproject vereist zijn, komen namelijk ook in een viewpoint-gebaseerde systeemontwikkeling methode terug.

Het grootste verschil ligt echter in het feit dat, wanneer men met viewpoints werkt er meer nadruk wordt gelegd op het controleren van de consistentie tussen modellen en er meer rekening wordt gehouden met stakeholder concerns. Wanneer alle modellen consistent zijn met elkaar hoeft dit niet altijd te betekenen dat het model ook betrouwbaar is. Als dezelfde fout overal in terugkomt lijkt het model namelijk ook (ten onrechte) correct. Hierdoor is moeilijk om te zeggen dat een systeemontwikkelingsproces dat gebaseerd is op het werken met viewpoints betrouwbaarder is, dan een systeem ontwikkelingsproces waar niet op basis van viewpoints wordt gewerkt.

Als men naar de voordelen en de mogelijkheden kijkt van het werken met viewpoint, neig ik meer te zeggen dat viewpoint-gebaseerde systeemontwikkeling vanwege de hogere mate van stakeholder betrokkenheid betrouwbaarder is. Momenteel zijn er echter te weinig empirische onderzoeken verricht om deze bewering te bevestigen of ontkennen.

7.5 Tool support

Er zijn nog geen goede commercieel elektronische hulpmiddelen beschikbaar die het werken met viewpoints en views volledig ondersteunen. Voor de acceptatie van viewpoint-gebaseerde systeemontwikkeling is het bestaan van tools die dit ondersteunen van essentieel belang. Zo’n hulpmiddel kan bijvoorbeeld de consistentie van verschillende views en het beheer van verschillende modellen ondersteunen. Het is belangrijk dat dit soort hulpmiddelen niet een bepaalde structuur opdringen ten koste van het leesgemak.

Hoewel er nog geen commerciële producten beschikbaar zijn, is men binnen de wetenschappelijke wereld druk bezig met experimenteren. Zo zijn er diverse viewpoints tools beschikbaar. Op de volgende pagina bevinden zich een tweetal voorbeelden.

- ViewPoint Tool
  De ViewPoint Tool [28] bestaat uit drie onderdelen, namelijk de ViewPoint Manager, Template Editor en de ViewPoint Editor. Deze tool is vooral bedoeld om inconsistenties tussen viewpoints te herkennen.

- The Viewer
  The viewer [28] is een prototype tool waarmee viewpoint-gebaseerde systeem ontwikkeling mogelijk is. Het is voornamelijk bedoeld om viewpoint-gebaseerde methodes te ontwikkelen en integratie ervan te ondersteunen en om viewpoints te maken en te manipuleren.

Het lijkt erop dat er nog een hele lange weg is te gaan voordat er bruikbare producten op de markt kunnen komen. Zonder goede tools acht ik de kans zeer klein dat systeemontwikkelaars aan viewpoint-gebaseerde systeemontwikkeling zullen doen. Zoals het nu eruit ziet zal de modelleur het voorlopig moeten doen met een tekstverwerker en een (UML-)modelleringprogramma.

7.6 Terugblik op het hoofdstuk

Het is gebleken dat wanneer men spreekt over het modelleren van viewpoints men eigenlijk het modelleren van views bedoelt. Een view is namelijk het resultaat van een viewpoint. Er bestaat een enorm arsenaal aan view modellen, viewpoints en gerelateerde views. Doordat er zoveel verschillende viewpoints bestaan is het niet altijd even gemakkelijk om de juiste viewpoint voor een specifiek situatie te kiezen. Als men bijvoorbeeld vanuit een functioneel gerichte perspectief modelleert dan zou het gebruik van Dataflow Diagrammen kunnen worden aangeraden. Als men bijvoorbeeld een organisatiestructuur of hiërarchie met de bijbehorende taken in kaart wil brengen is het gebruik van een organogram al voldoende. Voor iedere viewpoint moet er dus naar een geschikte taal worden gekeken.

Viewpoints zijn ervoor bedoeld om stakeholder perspectieven beter te kunnen structuren, organiseren en te managen. Het werken met viewpoints kan gezien worden als een middel om met het meerdere perspectieven probleem om te kunnen gaan. Daarnaast kunnen viewpoints als een soort handleiding worden beschouwd om tot een view te komen.

Momenteel bestaan er verschillende beschrijvingen van het begrip “viewpoint”. Zo hanteren diverse wetenschappers en wetenschappelijke organisaties in meer of mindere mate een eigen methodiek en terminologie ten aanzien van het werken met viewpoints.

Als gevolg hiervan zijn er door de loop der jaren tientallen verschillende view modellen en viewpoints ontwikkeld. Er wordt een breed scala aan opties aangeboden. Hoewel dit als een voordeel gezien kan worden, ervaar ik dit echter niet zo. Want hoe bepaal je nu welke view model het best geschikt is voor een architectuur of systeem. Standaarden als IEEE 1471 en ISO 10746 bieden de IT-architect, systeemontwikkelaar of projectmanager amper aanwijzingen of ondersteuning hierin. Het grote aanbod aan view modellen en het verschil in werkmethodiek en terminologie maakt kiezen van een view model niet eenvoudig. Het gebruik van een classificatie raamwerk zoals die door de Telematica Instituut is gepresenteerd bied ondersteuning bij het maken van een keuze.

Deel III

Data analyse

8. In hoeverre zijn viewpoints vernieuwend

Tijdens het bestuderen van de literatuur omtrent het werken met viewpoints ben ik zowel nieuwe als oude aspecten tegengekomen. Vaak doen schrijvers het laten voorkomen dat ze met iets totaal revolutionairs zijn gekomen. Als onderzoeker is het mijn taak om kritisch te zijn. Zo ben ik gaan analyseren in hoeverre het werken aan de hand van viewpoints vernieuwend is in de systeemontwikkeling.

John Zachman was een van de eerste personen die in 1987 het werken middels perspectieven aan het grote publiek introduceerde. Zijn ideeën over architecturen werden door veel mensen in de IT-wereld als vernieuwend en verfrissend gezien. Mede op basis van zijn werk zijn er door de loop der jaren tientallen verschillende view modellen en nieuwe viewpoint theorieën gecreëerd.

Systeemontwikkeling bestaat bijna net zo lang als dat de computer zelf bestaat. Het vormt één van de oudste en meest ontwikkelde vakgebieden binnen de automatiseringbranche. Natuurlijk worden er regelmatig nieuwe methoden bedacht of worden oude methoden verbeterd, maar een compleet vernieuwende concept voor het ontwikkelen van systemen komt niet zo vaak voor. Ik was dan ook zeer nieuwsgierig in hoeverre viewpoints de systeemontwikkeling konden vernieuwen en verbeteren.

Hoe langer ik met het onderzoek bezig was en meer te weten kwam over viewpoints, des te meer ik begon te twijfelen of het vernieuwende aspect ervan. Het ontwikkelen van systemen aan de hand van viewpoints is gebaseerd op het principe om vanuit de concerns en perspectief van de stakeholder te werken. Dit is toch iets wat altijd al zo is geweest?

Een computersysteem is namelijk bijna altijd voor gebruikers (of andere stakeholders) bestemd, systemen worden ontwikkeld om aan hun behoeften te voldoen. Als ontwikkelaar heb je dan ook eigenlijk geen andere keus dan rekening te houden met de behoeften van de toekomstige gebruikers. Je werkt (vaak onbewust) vanuit het perspectief van de gebruiker. Wanneer ontwikkelaars dit niet doen zal het eindresultaat hoogstwaarschijnlijk niet door de opdrachtgever worden geaccepteerd.

Eigenlijk bestaat de stakeholder al sinds het begin van de systeemontwikkeling, alleen werden ze toen nog niet expliciet vermeld. Het werken vanuit het belang van de belangrijkste stakeholders is dus eigenlijk iets dat (bewust of onbewust) van oudsher wordt gedaan. Ik ben er wel mee eens dat dit in het verleden niet altijd even professioneel en gestructureerd gedaan werd als nu het geval is.

In het begin van het computertijdperk stond de gebruiker dan ook aanzienlijk minder centraal dan nu. Daarom is het werken met viewpoints aan de ene kant zeker vernieuwend. Was men voor het viewpoint tijdperk onbewust bezig met perspectieven en concerns van stakeholders, is men nu juist zeer bewust met deze aspecten bezig.

Het werken met viewpoints dient naar mijn mening dan ook niet als iets revolutionair nieuws worden gezien, maar vooral als een toevoeging op de huidige manier van systeemontwikkeling.

9. Conclusie

9.1 Beantwoording van de deelvragen

Hieronder worden de deelvragen die aan het begin van deze afstudeerscriptie gedefinieerd werden beantwoordt.

9.1.1 Waarom falen veel ICT-projecten?

Er is geen eenduidige oorzaak voor het falen van ICT-projecten. Per situatie en per ICT-project kan dit namelijk verschillen en is men afhankelijk van een breed scala aan factoren. Het aantal factoren dat mogelijk van invloed kan zijn op het resultaat van een ICT-project is als het ware onbeperkt, waardoor het voor veel projectmanagers moeilijk is om projecten succesvol te kunnen managen.

Het zoeken naar specifieke oorzaken voor het falen van projecten is niet eenvoudig. Dit komt niet alleen doordat meerdere factoren van invloed kunnen zijn op het slagen of falen van ICT-project, maar ook doordat factoren elkaar onderling zowel direct als op indirecte wijze kunnen beïnvloeden.

Het achterhalen van de oorzaak waardoor een ICT-project als niet (of minder) succesvol wordt ervaren is daarentegen wel een stuk eenvoudiger te achterhalen. Onderzoek [16] heeft uitgewezen dat de voornaamste reden waarom veel ICT-projecten niet als succesvol worden ervaren, komt doordat de klant vindt dat het eindresultaat niet aan hun verwachtingen voldoet. Dit betekent dat een opdrachtgever teleurgesteld is in het opgeleverde product.

De volgende stap is het achterhalen van de reden waarom producten niet aan klantverwachtingen voldoen. We verkleinen hiermee de scope en verminderen zo het aantal factoren waarmee men rekening dient te houden.

Binnen de systeemontwikkeling vormt de requirements engineering het belangrijkste middel om de eisen, wensen en verwachtingen ten aanzien van het te ontwikkelen systeem te kunnen achterhalen. Het moge duidelijk zijn dat wanneer de requirements engineering te kort schiet, een ontwikkelaar geen duidelijk (of betrouwbaar) beeld krijgt van hoe het systeem eruit moet zien en welke functionaliteiten deze dient te bevatten. Een incomplete requirements engineering zal ongetwijfeld niet bijdragen tot het leveren van producten die beter aan de klantverwachtingen voldoen, maar zal eerder een averechts effect hebben.

Slechte communicatie tussen betrokkenen vormt een ander belangrijk oorzaak voor het niet voldoen aan klantverwachtingen. Het leveren van een product wat niet aan de verwachtingen voldoet is vaak een direct gevolg van het feit dat mensen elkaar vaak niet juist begrijpen. Geen enkel (oprechte) systeemontwikkelaar zal namelijk opzettelijk een verkeerde systeem bouwen. Verschillende onderzoeken [27, 9] hebben aangewezen dat een gebrek aan goede communicatie één van de belangrijkste oorzakelijke factoren vormt voor het falen van ICT-projecten. Overigs is voor het verrichten van een complete requirements engineering ook goede communicatie tussen betrokkenen nodig, aangezien deze twee factoren sterk aan elkaar gerelateerd zijn.

9.1.2 Wat zijn viewpoints?

Viewpoints zijn bedoeld als een manier om op bepaalde aspecten van een architectuur (of systeem) te kunnen focussen. Deze aspecten worden bepaald door de concerns van een stakeholder. Wat wel en niet zichtbaar moet zijn vanuit een bepaald viewpoint is daarom helemaal afhankelijk van de concerns van een stakeholder.

Iedere stakeholder heeft afhankelijk van zijn of haar taken, verantwoordelijkheden, kennis en evaringen(en), een uniek perspectief op het te ontwikkelen systeem. Hoe meer perspectieven er zijn waarmee rekening gehouden dient te worden, des te ingewikkelder het voor systeemontwikkelaars wordt om met elkaar te kunnen communiceren, de werkzaamheden te coördineren en een resultaat op te leveren dat aan de klantverwachtingen voldoet. Dit fenomeen staat ook wel bekend als het meerdere perspectieven probleem.

Het werken aan de hand van viewpoints is een manier om stakeholder perspectieven beter te kunnen structuren, organiseren en managen. Viewpoints kunnen hierdoor gezien worden als een hulpmiddel wat het mogelijk maakt om met het meerdere perspectieven probleem om te kunnen gaan. Hiernaast kan een viewpoint ook als een soort handleiding worden gezien om tot een view te komen.

Er zijn verschillende beschrijvingen van het begrip “viewpoint” in de omloop. Welke componenten een viewpoint-specificatie dient te bevatten is afhankelijk van de gekozen beschrijving. Hoewel er verschillende beschrijvingen zijn, blijft het principe met betrekking tot het werken met viewpoints echter hetzelfde. Men werkt namelijk vanuit het concern van de relevante stakeholder. Toch moet ik concluderen dat de beschikbaarheid van vele view modellen en viewpoints het maken van een keuze niet echt vergemakkelijkt en dat dit eerder een verwarrende dan verhelderende werking op een persoon kan hebben.

9.1.3 Kunnen viewpoints gemodelleerd worden?

Vanuit een praktisch oogpunt gezien heeft het weinig nut om viewpoints te modelleren, aangezien een viewpoint een soort handleiding is om tot een view te kunnen komen. Het is veel interessanter om views te modelleren. Een view is namelijk het resultaat van een viewpoint. Er bestaan tientallen verschillende viewpoints en ieder viewpoint beschikt daarnaast over een view. Omdat er enorm veel viewpoints bestaan is het niet altijd even eenvoudig om een gepaste viewpoint voor een bepaalde situatie te vinden.

Zowel binnen IEEE 1471 als in de meeste andere view modellen wordt er niet of nauwelijks beschreven hoe men een viewpoint moet kiezen en welke viewpoints men in bepaalde situaties dient te gebruiken. Men blijft over het algemeen nogal vaag over dit onderwerp, iets wat enigszins te maken zou kunnen hebben met feit dat er geen wereldwijd geaccepteerde standaard of werkwijze is. Ik kreeg (misschien ten onrechte) zo nu en dan sterk de indruk dat niemand concrete uitspraken durft te doen. Het komt natuurlijk niet echt professioneel over wanneer partijen elkaar zouden spreken. Dit ervaar ik als groot minpunt, want het maakt gebruik van viewpoint-gebaseerde systeemontwikkeling methoden niet aantrekkelijk voor het gebruik in de praktijk..

Uiteindelijk dient er toch een viewpoint gekozen te worden. Wanneer men eenmaal weet vanuit welk viewpoint er naar een bepaald domein gekeken moet worden, ligt de volgende uitdaging in het vinden van een geschikte modelleertechniek of -taal. Dit is zeker geen eenvoudige taak.. Er zijn namelijk tientallen verschillende view modellen en viewpoints mogelijk. Omdat er te veel verschillende opties zijn, kan men soms door bomen het bos haast niet meer zien. De mate van ervaring, vaardigheden en kennis die een ontwikkelaar tot zijn beschikking heeft, speelt daarom een belangrijke rol bij het maken van een viewpoint keuze. In een viewpoint wordt er tevens ook verteld hoe en met welke modelleertechnieken een view gemaakt moet worden. Aan de hand van deze informatie kan ook een ander persoon dan degene die de viewpoint heeft gemaakt, de gerelateerde view modelleren.

Omdat een view vanuit het perspectief van een stakeholder is gemaakt, kan deze ook aan de desbetreffende stakeholder gepresenteerd worden. Zodat deze persoon de betrouwbaarheid van het model kan verifiëren. Het controleren van views op fouten en inconsistenties vormt één van de belangrijkste voordelen van het werken met viewpoints.

Deze handelingen vergroten namelijk de kwaliteit van de views en helpen de ontwikkelaars of IT-architect het te onderzoeken domein beter te begrijpen.

9.1.4 In hoeverre is het werken met viewpoints betrouwbaar?

Stakeholder concerns kunnen door externe factoren beïnvloed worden. Een view wordt aan de hand van concerns opgesteld. Wanneer tijdens de ontwikkeling van een systeem, een stakeholder concern veranderd dient ook de view veranderd te worden om zo de correctheid ervan te kunnen waarborgen.

Op deze manier blijft de view representatief voor het perspectief van de stakeholder en blijft het de relevante concerns dekken. Een accuraat model komt natuurlijk ten goede aan de betrouwbaarheid van het model. Aan de andere kant kan er worden gezegd dat het werken met viewpoints werken juist niet betrouwbaard is, omdat views subjectief zijn. De consistentie controles kijken alleen of een model fouten bevat. Een foutloos model hoeft natuurlijk niet te betekenen dat het model correct is.

Hoe dan ook, zal het leren kennen van stakeholders en hun concerns ongetwijfeld leiden tot een beter begrip van het domein. Wanneer men (eenvoudige) modellen creëert die ook voor de stakeholder begrijpelijk zijn, is het mogelijk om over deze modellen met elkaar te communiceren. Samen in overleg met de stakeholder is het dan mogelijk de betrouwbaarheid van het model te verifiëren. Zo kan de stakeholder zelf zien of de voor hem of haar relevante belangen door het model gedekt worden. Indien nodig kan het één en ander eventueel worden aangepast.

Desalniettemin is het moeilijk om te zeggen of viewpoint-gebaseerde systeemontwikkeling methoden betrouwbaarder zijn dan traditionele systeemontwikkelingmethoden. Als men naar de voordelen en de mogelijkheden kijkt van het werken met viewpoints, neig ik meer te zeggen dat viewpoint-gebaseerde systeemontwikkeling vanwege de hogere mate van stakeholder betrokkenheid betrouwbaarder is

9.1.5 In hoeverre leidt het werken met viewpoints tot betere requirement engineering?

Een zeer belangrijke randvoorwaarde om computersystemen beter aan klantverwachtingen te laten voldoen is dat er een goede (complete) requirements engineering verricht dient te worden. Requirements horen de behoeften van stakeholders te weerspiegelen. Wanneer er geen goede requirements engineering wordt verricht en het hierdoor niet duidelijk is wat stakeholders willen, loopt de ontwikkelaar het risico een resultaat op te leveren dat deels of helemaal niet aan de klantverwachtingen voldoet.

Hoe komt het dat stakeholderverwachtingen met betrekking tot een systeem per persoon verschillend kunnen zijn? Dit komt doordat iedere stakeholder een eigen (uniek) perspectief (kijk) op de wereld heeft. Deze perspectieven wordt aan de hand van de ervaring(en), kennis, taken en verantwoordelijkheden van een persoon gevormd. Op grond van deze aspecten hebben stakeholders één of meerdere concerns ten opzichte van het systeem. Een systeem zal in de ogen van de stakeholder pas goed zijn als deze aan hun concerns voldoet. De kunst van het verrichten van een goede requirements engineering is het onder andere kunnen vinden van verborgen requirements.

Vaak kunnen stakeholders hun eisen en wensen met betrekking tot het nog te ontwikkelen systeem niet duidelijk aangeven. Dit komt doordat ze een deel van de requirements als vanzelfsprekend beschouwen. Zo gaan de meeste mensen bij het kopen van een auto er automatisch vanuit dat deze een stuur en vier wielen heeft. Weinig mensen zullen deze (verborgen) requirements op hun verlanglijstje zetten. Als men een auto koopt, stelt men in de meeste gevallen hooguit een aantal eisen met betrekking tot de kleur, opties en accessoires. Zo gaat dat met computersystemen ook. Aspecten die door een gebruiker onbewust als vanzelfsprekend worden beschouwd, zullen ongetwijfeld door een stakeholder niet expliciet beschreven worden. Echter, wanneer de requirements engineer tijdens de ontwikkeling van de auto deze (verborgen) requirements niet ontdekt, zal de auto bij levering geheid niet aan de klantverwachtingen voldoen. Een excuus zoals “u heeft aanvankelijk bij het vermelden van uw eisen en wensen niets over een stuur en wielen gezegd” zal zeer waarschijnlijk niet door klant geaccepteerd worden, waardoor de koop niet zal doorgaan.

Met computersystemen werkt het ook net zo. De requirements engineer probeert via alle relevante stakeholders de requirements voor het systeem te achterhalen. Omdat men de requirements engineer van te voren niet weet welke requirements benodigd zijn, kan het achterhalen van requirements een heel complex en langdurig proces zijn.

In veel gevallen kunnen toekomstige gebruikers niet concretiseren wat ze precies willen wat het systeem doet. Vaak hebben ze wel een idee, maar zijn ze niet in staat om dit helder en tijdig door te communiceren aan de ontwikkelaars. Naar mate de ontwikkeling van een systeem verder vordert, zullen in de meeste gevallen de stakeholders hun eisen en wensen ten aanzien van het systeem steeds beter kunnen verduidelijken. Dit is natuurlijk geen ideaal scenario, want hoe verder een project vordert, des te minder het gewenst is om nieuwe requirements te ontvangen en deze requirements te implementeren in het systeem. Zo zullen hierdoor niet alleen zullen de kosten van het project stijgen, maar zal ook het project langer gaan duren dan oorspronkelijk gepland was. Daarom is het

belangrijk dat requirements tijdig worden achterhaald en dat deze bovendien ook correct zijn. Op deze manier kunnen mogelijk resources bespaard worden.

Ik ben van mening dat het werken aan de hand van viewpoints de requirements engineering zal verbeteren. Dit baseer ik op het feit dat iedere stakeholders concerns heeft. Wanneer een ontwikkelaar wil dat het te ontwikkelen systeem aan de klantverwachtingen voldoet, dient dit systeem natuurlijk aan de concerns van de klant te voldoen.

Stakeholder concerns vormen een essentieel onderdeel bij het werken met viewpoints. Zo kent ieder viewpoint het component “concerns”. Hierin worden de stakeholder concerns beschreven. Zonder stakeholder concerns is een viewpoint geen viewpoint. Stakeholder concerns zijn echter niet hetzelfde als systeem requirements. Men kan namelijk concerns niet klakkeloos overnemen en dan vervolgens als systeem requirements definiëren. Het is echter wel een stap in de goede richting. Zo is het mogelijk dat ontwikkelaars aan de hand van de concerns kunnen zien welke aspecten belangrijk zijn voor een bepaalde stakeholder. Wanneer men eenmaal weet welke concerns voor een stakeholder van belang zijn, kan de requirements engineer alle aspecten achterhalen waaraan het systeem moet voldoen om aan deze specifieke concerns te voldoen.

Zelfs wanneer een stakeholder (bijvoorbeeld een eindgebruiker) denkt dat hij of zij alle relevante requirements voor een systeem heeft verteld, kan de requirements engineer aan de hand van de concerns controleren of dit daadwerkelijk het geval is. Zo kan de requirements engineer zelf op basis van de gegeven concerns onderzoeken of er nog additionele requirements bedacht kunnen worden die een bijdrage kunnen leveren aan de ondersteuning van deze concerns.

Viewpoints kunnen gezien worden als een middel om de requirements engineering completer te maken. Als de requirements engineer zelf op basis van de stakeholder concerns, veel nieuwe requirements kan bedenken die aansluiten op concerns, kan men min of meer concluderen dat de initieel verrichte requirements engineering niet compleet was. Wanneer er niet of nauwelijks nieuwe requirements bedacht kunnen worden, kan men er vanuit gaan dat de huidige requirements voldoende zijn om de concerns te dekken en kan men concluderen dat de requirements engineering redelijk compleet is.

View modellen leveren een uitgedachte methodiek om stakeholder views te kunnen modelleren. Het hanteren van een goed doordachte werkmethodiek zal de kwaliteit van views verbeteren en zal de kans doen afnemen dat er belangrijke aspecten over het hoofd worden gezien. Wanneer men meerdere view-representaties heeft en de situatie vanuit het perspectief van een stakeholder bekijkt, zal de requirements engineer een beter inzicht krijgen in het domein. Als men de situatie vanuit het perspectief van de klant bekijkt, wordt het een stuk eenvoudiger om zich voor te stellen wat wel of niet relevant is voor deze stakeholder.

Het gebruik van viewpoints garandeert natuurlijk niet dat een resultaat aan alle klantverwachtingen voldoet, maar het vergroot in bepaalde mate zeker de kans dat de

requirements engineering completer kan worden verricht. Hierdoor neem ook de waarschijnlijkheid toe dat producten beter aan klantverwachtingen voldoen.

9.1.6 In hoeverre verbeteren viewpoints de communicatie tussen betrokkenen?

Het werken aan de hand van viewpoints zal (mits het proces goed gemanaged wordt) de communicatie tussen stakeholders ondersteunen of zelfs bevorderen. Communicatie is een essentiële factor voor het doen slagen of falen van een ICT-project. Slechte communicatie tussen betrokkenen kan er namelijk toe leiden dat er verkeerde keuzes gemaakt worden en het project uiteindelijk mislukt. Daarom is het belangrijk dat er duidelijke afspraken, richtlijnen en werkprocedures worden opgesteld. Op deze manier kan het communicatieproces binnen een project beter beheerst worden.

Hoe meer mensen er bij een project betrokken zijn, des te moeilijker het voor ze wordt om goed en in het bijzonder duidelijk met elkaar te kunnen communiceren. Een concreet voorbeeld hiervan is het meerdere perspectieven probleem dat in paragraaf 7.1.1 is behandeld. Zo kunnen diverse projectmedewerkers (afhankelijk van hun expertise) verschillende modelleertechnieken gebruiken om de voor hun relevante views te kunnen modelleren. Als er binnen een projectgroep verschillende modelleertalen worden gehanteerd, wordt het een stuk complexer om deze modellen op consistentiefouten te controleren. Voor een uitgebreidere beschrijving van het meerdere perspectieven probleem wordt verwezen naar paragraaf 7.1.1 en 7.1.2.

Het meerdere perspectieven probleem dient dus vooral als een communicatief probleem beschouwd te worden. Viewpoints zijn bedoeld als een hulpmiddel om met het meerdere perspectieven probleem om te kunnen gaan en deze te managen. Dit laat zien dat viewpoints in ieder geval als doelstelling hebben de communicatie tussen betrokkenen te verbeteren.

Afhankelijk van de viewpoint-specificatie die men hanteert (IEEE 1471 of een ander), bevat iedere viewpoint een component waarin de te gebruiken modelleertaal staat voorgeschreven en hoe de gerelateerde view dient te worden opgebouwd. Een viewpoint is dan vaak ook een soort handleiding voor de ontwikkelaar om views te kunnen modelleren. Als een projectmedewerker, door wat voor een reden dan ook, vroegtijdig met het project stopt, kan een ander plaatsvervangend projectmedewerker aan de hand een viewpoint het werk verder afmaken.

Viewpoints dwingen de ontwikkelaar of IT-architect om over alle viewpoint-componenten na te denken. Zo moeten onder andere de componenten “stakeholders”, “concerns” en “modelleertaal” beschreven worden. Om deze informatie te kunnen achterhalen, is men genoodzaakt om met stakeholders te communiceren. Zonder de informatie die door de stakeholders wordt verstrekt, zal men namelijk niet in staat zijn om de viewpoint-componenten te beschrijven. Deze behoefde om contact te onderhouden tussen ontwikkelaars en stakeholders zal het communicatieproces gaande houden

Door de stakeholder concerns te achterhalen en analyseren, leert men het domein beter kennen, wat weer als gevolg heeft dat de ontwikkelaar betere vragen kan stellen en een product op kan leveren wat beter aan de verwachtingen voldoet.

Het gebruik van viewpoints heeft dus zeker communicatieve voordelen. Er is echter wel één mogelijke belemmering dat zich voordoet wanneer er sprake is van tientallen verschillende stakeholders en viewpoints. Hoe meer viewpoints er zijn waarmee er rekening gehouden dient te houden, des te onoverzichtelijker het wordt om het communicatieproces goed te managen. Een beperkt aantal viewpoints en views op inconsistenties controleren gaat nog wel, maar wanneer er tientallen verschillende viewpoints zijn, wordt het communicatie proces een stuk complexer.

Bij grote projecten vrees ik dan ook dat het werken aan de hand van viewpoints eerder verwarring en stress zal opwekken, dan dat er orde en overzicht wordt gecreëerd. De projectmanager dient continu de aandacht erbij te houden om alles in goede banen te kunnen geleiden. Het gebruik van viewpoints is tevens arbeidsintensief, omdat het geen eenvoudige handelingen zijn die je even gauw in een korte periode kan verrichten.

Een ander sterk punt van het werken met viewpoints, is dat men een systeem vanuit verschillende perspectieven kan bekijken. Hierdoor kan een ontwikkelaar relevantere vragen stellen aan stakeholders. Dit betekent dat ontwikkelaars het domein beter zullen leren begrijpen en de stakeholders beter zullen leren kennen. Aan de hand van deze informatie zullen ze in staat zijn om realistischere modellen te maken. Middels deze modellen kunnen de ontwikkelaars weer vervolgens beter met de overige stakeholders communiceren.

De mate in hoeverre viewpoints bijdragen tot een verbetering of ondersteuning van de communicatie is afhankelijk van de complexiteit van het te onderzoeken domein of te ontwikkelen systeem. Hoe groter het domein of systeem, des te meer mensen er betrokken zijn en des te meer perspectieven er ontstaan. Veel perspectieven leiden tot veel viewpoints en veel viewpoints kunnen het structureren en werken met viewpoints onoverzichtelijk maken.

9.2 Beantwoording van de centrale onderzoeksvraag

De voornaamste reden waarom veel ICT-projecten als niet of minder succesvol wordt ervaren komt doordat opgeleverde systemen niet aan klantverwachtingen voldoen. Dit valt voornamelijk te verwijten aan een incomplete requirements engineering en/of slechte communicatie tussen betrokkenen.

Als men ervoor wil zorgen dat systemen beter aan de klantverwachtingen voldoen, is het belangrijk om de requirements engineering en de communicatie tussen de betrokkenen te verbeteren.

Het werken aan de hand van “de juiste” viewpoints zal de communicatie tussen betrokken verbeteren en zo uiteindelijk ook tot een completere requirements engineering leiden. Het is jammer dat klantverwachtingen niet objectief gemeten kunnen worden. Dit komt doordat de manier waarop de mens de realiteit waarneemt voor een groot deel met sociale en subjectieve aspecten te maken heeft. Wat voor de ene stakeholder goed kan zijn, hoeft voor de andere stakeholder dat niet te zijn. Men loopt altijd het risico om niet aan alle stakeholderverwachtingen te voldoen. Het probleem met mensen is, dat hun gedachten niet statisch zijn waardoor requirements door de loop van een project kunnen

veranderen. In de beginfase is het nog relatief eenvoudig om op requirements-wijzigingen te anticiperen, maar hoe verder men in het project is gevorderd des te moeilijker dit wordt. Op een gegeven moment bereikt men een fase waarin wijzigingen niet meer kunnen worden doorgevoerd. Het is vanaf dit cruciale punt dat een product wel of niet meer kan voldoen aan klantverwachtingen.

Om deze reden dient de ontwikkelingstijd van een systeem zo kort mogelijk gehouden te worden en is het verstandig om een systeem (indien mogelijk) modulair te ontwerpen, zodat deze in de toekomst eventueel geüpgrade kan worden. Om systemen sneller te kunnen ontwikkelen bestaan er zogenaamde RAD (Rapid Application Development) [47] en DSDM (Dynamic System Development Methodology) [7] methodes, die erop gericht zijn om zo snel mogelijk systemen te kunnen ontwikkelen. Viewpoint-gebaseerde systeemontwikkelingmethoden daarentegen, zullen het ontwikkelingsproces juist verlengen. Zo moeten stakeholder-profielmodellen en viewpoints worden opgesteld en dienen de gerelateerde views gemodelleerd te worden. Uiteindelijk zullen deze views met elkaar vergeleken moeten worden en zullen consistentiefouten eruit moeten worden gefilterd. Deze handelingen dienen voor de realisatiefase van een systeemontwikkeling traject te worden uitgevoerd.

Ook het feit dat er maar in beperkte mate literatuur aanwezig is over hoe men viewpoints dient te kiezen of te maken, komt niet ten goede aan de duur van een project. Het gebrek aan een duidelijke methodiekbeschrijving vergroot de kans dat er fouten gemaakt worden en kan ertoe leiden dat producten toch uiteindelijk niet aan klantverwachtingen voldoen.

Bij viewpoint-gebaseerde systeemontwikkelingtrajecten zal de beginfase (in vergelijking tot traditionele methoden) dan ook wat langer duren. Echter, zal mede door een verbeterde requirements engineering de kans vergroot worden dat het product meer aan de klantbehoeften voldoet. Op langere termijn zal een viewpoint-gebaseerde aanpak (mits deze correct is uitgevoerd) zijn vruchten afwerpen.

Er zijn een aantal voorwaarden om aan de hand van viewpoints producten op te kunnen leveren die beter aan klantverwachtingen voldoen. Allereerst is het belangrijk dat viewpoint concerns altijd aan de betreffende stakeholders terug gekoppeld worden. Zo kunnen stakeholders zelf aan de systeemontwikkelaar of IT-architect vertellen of deze accuraat zijn of niet. Door met elkaar te communiceren, kan een ontwikkelaar meer inzicht verkrijgen in het perspectief van de stakeholder. Het hebben van correcte concerns vormt de basis voor het opzetten van een viewpoint. Wanneer een viewpoint verkeerd is kan dit nadelige gevolgen hebben voor het opleveren van producten die aan de verwachtingen van stakeholders voldoen.

Een ander voorwaarde is dat men zich moet realiseren dat het haast onmogelijk is om aan alle stakeholder concerns te voldoen. Vooral bij grote projecten, waar men met veel stakeholders te maken heeft zal men bereid moeten zijn om concessies te doen. Als dit niet wordt gedaan kan dit leiden tot ingewikkelde onderhandelingen die de duur van het project aanzienlijk kunnen verlengen. Men doet er hier verstandig aan om te achterhalen welke stakeholders de belangrijkste zijn en ze vervolgens te rangschikken.

Bij het achterhalen van concerns heeft men (in min of meerdere mate) te maken met dezelfde problematiek als met de requirements engineering. Hoe weet men namelijk of alle relevante concerns zijn achterhaald? Dit is heel moeilijk, aangezien er geen eenvoudige analytische procedure bestaat om te bepalen of stakeholders de systeemontwikkelaar of IT-architect alles hebben verteld wat ze nodig hebben om de viewpoint en de gerelateerde view te kunnen maken. Het vinden van concerns is essentieel bij een viewpoint-gebaseerde aanpak. Het verbaast me dan ook enigszins dat er geen concrete methodiek voor handen is om stakeholder-profielmodellen te creëren.

Klanten komen vaak na de implementatie van een systeem tot de conclusie dat deze niet aan hun verwachtingen voldoet. De mate van klanttevredenheid hangt in grote mate af van in hoeverre een viewpoint de stakeholder concerns dekt. Ook dient men er serieus rekening mee te houden dat stakeholders zich niet altijd bewust zijn van concerns. Deze "hidden concerns" kunnen doorslaggevend zijn voor het al dan niet voldoen aan klantverwachtingen en vormen daarom een zeer belangrijke voorwaarde om beter aan stakeholderverwachtingen te voldoen.

Ten slotte dient een viewpoint gedurende het project geldig te zijn. Als concerns veranderen en een viewpoint niet wordt aangepast, verliest de bijbehorende view zijn waarde. Deze is namelijk dan niet meer representatief voor de huidige realiteit. IEEE 1471 kent momenteel geen minimale geldigheideisen voor een viewpoint. Het toevoegen van minimale viewpoint-eisen zou het eenvoudiger maken om de geldigheid van een viewpoint te controleren en beheersen.

" In hoeverre leidt het ontwikkelen van systemen aan de hand van viewpoints tot systemen die beter aan klantverwachtingen voldoen? "

Er bestaat geen enkele methodiek die 100% kan garanderen dat men aan alle klantverwachtingen voldoet, aangezien de menselijke geest hier gewoonweg te complex, dynamisch en subjectief voor is. Toch zijn er genoeg aanwijzingen te vinden dat het werken met viewpoints een positieve bijdrage zal hebben op de systeemontwikkeling. Bij viewpoint-gebaseerde systeemontwikkeling draait alles namelijk om de stakeholder.

Wanneer projectmanagers, systeemontwikkelaars of IT-architecten rekeninghoudend met de bovenstaande voorwaarden, systemen vanuit het perspectief van de meest belangrijke stakeholders ontwikkelen, zal dit ongetwijfeld de waarschijnlijkheid verhogen dat opgeleverde systemen beter aan de klantverwachtingen voldoen. Het ontbreken van voldoende praktijkervaring maakt het echter momenteel onmogelijk om deze bewering onweerlegbaar te bewijzen.

Vanzelfsprekend kwam ik dus tot de volgende vraagstelling "Waarom worden viewpoint-gebaseerde systeemontwikkelingmethoden momenteel zo weinig toegepast in de praktijk?". Literatuur omtrent het werken met viewpoints bestaat al sinds de jaren tachtig. Men zou denken dat er inmiddels een breed scala aan systeemontwikkelingsprojecten zou zijn verricht op basis van viewpoints. Om een wetenschappelijke verklaring voor deze vraagstelling te kunnen verkrijgen zou er eigenlijk nog een vervolgonderzoek moeten worden verricht, iets wat ik ook in de volgende paragraaf adviseer.

Op basis van de kennis en ervaring die ik aan de hand van dit onderzoek heb opgedaan kan wellicht een mogelijke verklaring geven voor dit fenomeen. Houdt er echter wel rekening mee dat deze verklaring nog bewezen dient te worden.

De reden dat viewpoint-gebaseerde systeemontwikkelingmethoden (in Nederland) relatief weinig toegepast worden in de praktijk, heeft volgens mij enigszins te maken met de vaagheid van de literatuur omtrent IT-architecturen. Zo ben ik tijdens mijn onderzoek regelmatig tegen wollig taalgebruik aangelopen. Daarnaast zijn er momenteel zoveel view modellen en viewpoints beschikbaar dat men door de bomen het bos haast niet meer ziet.

Het feit dat de terminologie nog niet gestandaardiseerd is, maakt het gebruik van viewpoint-gebaseerde systeemontwikkeling ook niet bepaald aantrekkelijk.. Wanneer bijvoorbeeld twee verschillende vestigingen van een bedrijf, verschillende view modellen gebruiken (waarbij verschillende termen voor de zelfde betekenissen gebruikt worden), zal de communicatie tussen deze twee vestigingen moeizaam verlopen.

Het ontbreken van een gestandaardiseerde methodiek ten aanzien van het werken met viewpoints gecombineerd met het feit dat het nogal veel kennis en inspanning kost om een viewpoint-gebaseerde aanpak te kunnen realiseren, betekent in veel gevallen dat het werken aan de hand van viewpoints niet weggelegd is voor kleine bedrijven. Vaak ontbreekt bij kleine bedrijven de resources om grote en complex ICT-projecten te verrichten. Het gebruik maken van viewpoint-gebaseerde systeemontwikkelingmethoden is simpelweg niet rendabel genoeg voor ze. Om deze reden zijn viewpoint-gebaseerde methoden meer geschikt voor grote multinationals. Boeing, Volkswagen, General Motors en Bank of America zijn bijvoorbeeld multinationals die op grote schaal het Zachman-raamwerk hebben toegepast. Ook in Nederland hebben de laatste jaren een aantal bedrijven zoals Akzo Nobel [48] en Atos Origin [49], weliswaar op een kleinere schaal en in beperktere mate, gebruik gemaakt van view modellen.

Enterprise architectuur denken is een fenomeen dat de afgelopen jaren sterk in populariteit is gegroeid. Waar sprake is van enterprise architecturen komt men ook het gebruik van viewpoints tegen. Architecturen zijn vaak complex en hebben met veel stakeholder perspectieven te maken. View modellen spelen bij het oplossen van architectuurvraagstukken een sleutelrol. Mijn verwachting is dan ook dat de aankomende jaren het werken aan de hand van viewpoints steeds noodzakelijk zal worden om dit complexiteit dat gepaard gaat met architecturen te kunnen beheersen.

9.3 Suggesties voor verder onderzoek

Naar aanleiding van dit onderzoek zou het interessant zijn om te kunnen achterhalen waarom viewpoint-gebaseerde systeemontwikkelingmethoden momenteel maar in beperkte mate worden toegepast in de praktijk. Het werken aan hand van viewpoints brengt namelijk een aantal grote voordelen met zich mee. Zo krijgen ontwikkelaars meer inzicht in het te onderzoeken domein, waardoor de kans wordt vergroot op het leveren van producten die aan beter aan stakeholderverwachtingen voldoen. Veel ICT-projecten mislukken namelijk doordat het opgeleverde eindresultaat niet aan de klantverwachtingen voldoet. Wanneer een ICT-project mislukt en het eindresultaat onbruikbaar is, gaan er logischerwijs resources verloren. Het werken aan de hand van viewpoints vermindert de

kans op het afleveren van producten de niet aan klant verwachtingen voldoen. Je zou denken projectmanagers, systeemontwikkelaars en IT-architecten zouden popelen voor een dergelijke methode. Er is vast een verklaring te vinden voor het feit dat er op zo'n beperkte schaal op basis van viewpoints wordt gewerkt.

Een ander interessant onderwerp om te onderzoeken zou kunnen zijn of het mogelijk is om een eenvoudige methodiek te ontwikkelen om viewpoints te kunnen kiezen, opstellen en gebruiken. Het grootste probleem dat ik heb ondervonden gedurende mijn onderzoek is dat er simpelweg te weinig informatie te vinden was over methoden om viewpoints te kiezen. Het hebben van een begrijpelijk methodiek vormt naar mijn mening een belangrijk voorwaarde om viewpoint-gebaseerde systeemontwikkeling in de praktijk toe te passen.

9.4 Persoonlijk evaluatie van het traject

Het verrichten van onderzoek was veel moeilijker dan ik vooraf had gedacht. Er wordt namelijk een hoge mate van zelfstandigheid en discipline van de onderzoeker vereist. Iets waar ik in het begin amper mee rekening had gehouden.

Hoe verder ik in het onderzoek gevorderd was, des te meer kennis ik vergaarde. Pagina's waarvan ik dacht dat ik ze had afgerond, bleken met de nieuwe kennis die ik verworven had achteraf beter geschreven kunnen worden. Met als gevolg dat ik het document telkens keer op keer opnieuw ging herschrijven.

Het moeilijkste onderdeel was voor mij dan ook de afronding van mijn scriptie. Er was namelijk zo veel informatie wat ik wou vertellen. Op een gegeven moment kreeg ik echt het gevoel dat er geen einde meer in zicht was, waardoor ik in een soort motivatie dip belandde. Het was pas na een goed gesprek met mijn afstudeerbegeleider waardoor ik tot inzien kwam dat men op een gegeven moment een punt achter moet zetten en dat perfectie hoewel men dat dient te streven vaak niet haalbaar is. Het kan namelijk altijd beter.

De afgelopen paar maanden heb ik als zeer leerzaam ervaren. Ik kijk dan ook met een goed gevoel terug. Niet alleen heb meer academisch leren denken, maar ook heb ik geleerd niet alles vanzelfsprekend te beschouwen en vooral kritisch tegen dingen aan te kijken.

Literatuur

[42] Afbeelding van zachman raamwerk, http://ea.oio.dk/andre-arkitekturmetoder/zachman/zachman, Bezocht op 22-10-2007

[5] Algemene Rekenkamer, Lessen uit ICT-projecten bij de overheid - deel A, Den Haag, 2007.

[49] Atos Origin, Architectuurstudie Digitaal Bouwloket, Utrecht, 2006.

[22] Beender, N., Succes en falen van ICT trajecten, Vrije Universiteit van Amsterdam, Amsterdam, 2004.

[9] B I S N E Z Management en Business & IT Trends Institute en Vrije Universiteit van Amsterdam, Projectmanagementenquête 2006/2007 - Handvatten voor succesvolle projecten, Bergambacht, 2007

[45] Bolognesi, T. en Brinksma, E., Introduction to the ISO Specification Language LOTOS, *Computer networks and ISDN Systems*, vol 14, 1988

[18] Bullen, C.V. en Rockart, J.F.A., primer on critical success factors, Massachusetts Institute of Technology, Cambridge, 1981

[19] Daniel, D.R., Management Information Crisis, *Harvard Business Review*, vol 39 no 5, 1961.

[47] Developers.net, Rapid Application Development, http://www.blueink.biz/RapidApplicationDevelopment.aspx, Bezocht op 6-10-2007

[29] Doest, H. der. en Lacob, M. en Lankhorst, M. en Leeuwen, D. van. en Slagter, R., Viewpoints Functionality and Examples, Telematica Instituut, Enschede, 2004

[7] DSM consortium, Over DSM "De Methode", http://www.dsdm.nl/over/methode/, bezocht op 02-09-2007

[52] Easterbrook, S. en Nuseibeh, B., Requirements Engineering: A Roadmap, *22nd International Conference on on Software Engineering*, Limerick, 2000.

[51] Elswijk, M. van en Maat, M. en Wijngaard, E., van der, IT-architecten blinken uit in onduidelijkheid, *Automatiseringsgids*, vol 37 no 38, 2003

[13] Encyclopædia Britannica, zoekquery "social science", http://www.britannica.com, bezocht op 13-12-2007

[4] Financieel Management, Mislukken ICT-projecten kost 1,7 miljard per jaar, http://www.financieel-management.nl/content/view/800, bezocht op 12-09-2007

[28] Finkelsetin, A. en Kramer, J. en Nuseibeh, B. en Finkelstein, L. en Goedicke, M., Viewpoints: A Framework for Integrating Multiple Perspectives in System Development, *International Journal of Software Engineering and Knowledge Engineering*, 1992

[20] Fui-Hoon Nah, F., Lee-Shang Lau, J., Kuang, J., Critical factors for successful implementation of enterprise systems, *Business Process Management Journal*, Vol 7 no 3, 2001.

[34] Greefhorst, D. en Koning, H., Dimensies in architectuur, *Landelijk Architectuur Congres*, 2002.

[3] Heur, R., van, Meer dan helft ict-projecten mislukt, *Computable*, http://www.computable.nl/nieuws.htm?id=2019446, Bezocht op 20-06-2007

[41] Heuvel, W.J., van den en Proper, H.A., De pragmatiek van architectuur, *Informatie* Vol 44, 2002.

[12] ICT~Office, en Helifax, ICT Marktmonitor 2007, Woerden, 2007.

[44] ISO, Reference Model for Open Distributed Processing, 1998

[15] IEEE Standards Department, The Architecture Working Group of the Software Engineering Committee, IEEE Recommended Practice for Architectural Description of Software-Intensive Systems, New York, 2000

[16] Ernst & Young, ICT Barometer 2007, Amsterdam, 2007.

[23] Jones, C., Software Project Management Practices: Failure Versus Success, Software Productivity Research LLC, Hendersonville, 2004.

[55] Koning, H., Inleiding Architectuurbeschrijving, *SERC, Landelijk architectuur congres*, 2002

[37] Kontio, M., Architectural manifesto: Designing software architectures, IBM, http://www.ibm.com/developerworks/wireless/library/wi-arch11/, Bezocht op 10-08-2007

[36] Kotonya, G. en Sommerville, I., Requirements Engineering With Viewpoints, Lancaster University, Lancaster, 1995

[24] Whittaker, B., "What Went Wrong? Unsuccessful Information Technology Projects", Information Management & Computer Security vol 7 no1, 1999.

[31] Kruchten, P., Architectural Blueprints - The "4+1" View Model of Software Architecture, Rational Software Corp., Vancouver, 1995

[33] Kudrass, T., Describing Architectures Using RM-ODP, UBS AG, Zürich, 1999

[8] Langendijk , S., Falend projectbeheer hindert sector, *Computable*, no 23, 2006

[30] Lassing, N. en Rijsenbrij, D. en Vliet, H., van, Zicht op Aanpasbaarheid, Vrije Universiteit van Amsterdam, Amsterdam, 2001.

[38] Miles, R., Answer:Draw activity diagram showing a blog review, http://www.umlranch.com/activity-diagram-answers/2006/7/11/answer-draw-an-activity-diagram-showing-a-blog-review-process.html ,UML Ranch, Bezocht op 10-08-2007

[21] Molen, H.J.K.W., van der., Projecten vroegtijdig evalueren vergroot kans op succes, *Computable*, Vol 25, 2000.

[54] Morsink, B., Raamwerken, http://www.dekkermorsink.nl/raamwerken.htm, Bezocht op 04-08-2007

[48] Nagtegaal, S. en Koopman, B., Akzo Nobel Internet Platform – Architectuur beschrijving, Akzo Nobel, Amsterdam, 2005.

[2] NetoaAlvarez, S.J., Project management failure main causes, Bowie State University, Bowie, 2003.

[10] Nuseibeh, B. en Easterbrook, S., Requirements Engineering: A Roadmap, http://www.doc.ic.ac.uk/~ban/pubs/sotar.re.pdf, University of Toronto, Toronto, 2000.

[50] Proper, H.A., Grounded Enterprise Modelling, Radboud Universiteit, Nijmegen 2007

[53] Quakernaat, J., Master thesis -Het analyseren en verbeteren van een architectuurbeschrijving, Universiteit van Amsterdam, Amsterdam, 2006

[14] Radboud Universiteit Nijmegen, Master Informatiekunde, http://www.ru.nl/aspx/get.aspx?xdl=/views/run/xdl/page&SitIdt=109&VarIdt=86&ItmIdt=8062, Bezocht op 11-09-2007.

[26] Reel, J.S., Critical Success Factors in Software Projects, IEEE *Software*, May/June 1999.

[11] Rippen, R., Ict groeit flink, *Computable*, no 3. 21-01-2005, http://www.computable.nl/artikel.jsp?rubriek=1272857&id=1426456, Bezocht op 23-10-2007

[32] Soni, S. en Nord R.L en Hofmeister, C., Software architecture in industrial applications, *17th International Conference on Software Engineering*, New York, 1995.

[46] Spivey, J.M., The Z-notation: A reference manual, University of Oxford, Oxfort, 1998

[27] Spikes Cavell, BULL Survey, London, 1998

[17] Standish Group, The Standish Group Chaos Report, Boston, 1995.

[25] Standish Group, The Standish Group Chaos Report, Boston, 2006.

[1] Young, R., For IT projects, Silence can be deadly, *Journyx Project Management Blog*, http://ravenyoung.spaces.live.com/blog/cns!17376F4C11A91E0E!3461.entry, Bezocht op 28-04-2007.

[39] Use case scenario ATM figuur, http://argouml-stats.tigris.org/documentation/printablehtml/manual/images/tutorial/atm_use_case_diagram_include.gif Bezocht op 10-08-2007

[43] Vallecillo, A.,RM-ODP: The ISO Reference Model for Open Distributed Processing, Universidad de Málaga, Malaga, 2000

[6] Veld, S. van 't, 16 Methoden voor systeemontwikkeling, een vergelijkend rapport van de NGGO, Tutein Nolthenius, Den Bosh, 1990.

[40] Zachman, J.A., A framework for information systems architecture. *IBM Systems Journal,* vol 26 no3, 1987.

Bijlage I -Viewpoint Meta-model