Katholieke Universiteit Nijmegen

ICT Complexiteit Binnen Organisaties
“Architectuur als stuurmiddel?”

Colofon

Auteur:	**Ing. Roel Konieczny**	**rkoniecz@sci.kun.nl**
Opleiding:	**Informatiekunde**	
Opdracht:	**ICT complexiteit binnen organisaties**	
Universiteit:	**Katholieke Universiteit Nijmegen**	
Subfaculteit:	**Nijmeegs Instituut voor Informatica en Informatiekunde (NIII)**	
Afstudeer Begeleider:	**Dr. Stijn Hoppenbrouwers**	**stijnh@cs.kun.nl**
Meelezer:	**Prof. Dr. Erik Proper**	**E.Proper@acm.org**
Plaats, datum:	**Nijmegen, 7 mei 2004**	

nijmeegs instituut voor informatica en informatiekunde

Bezoekadres:	**Toernooiveld 1 6525 ED Nijmegen**
Postadres:	**Postbus 9010 6500 GL Nijmegen**
Telefoon:	**(024) 365 20 84**
E-mail:	**info@cs.kun.nl**

Voorwoord

In het kader van mijn studie Informatiekunde aan de Katholieke Universiteit Nijmegen heb ik een afstudeeronderzoek uitgevoerd bij de onderzoeksgroep Informatie en Kennissystemen (IRIS) binnen de subfaculteit Nijmeegs Instituut voor Informatica en Informatiekunde (NIII). Dit onderzoek vond plaats in de periode van 1 november 2003 tot en met 7 mei 2004.

Ik kijk met veel plezier terug op deze afstudeerperiode. Het onderzoek was erg vernieuwend en interessant. Ik zal deze periode altijd als waardevol beschouwen.

Speciaal wil ik hierbij mijn begeleider, Stijn Hoppenbrouwers, noemen. Hij heeft mij op een goede en positieve manier ondersteuning en feedback gegeven. Daarnaast wil ik Erik Proper bedanken voor de goede ondersteuning tijdens de uitvoering van het afstudeerproject.
Als laatste wil ik Martijn Scheepstra bedanken voor de goede samenwerking tijdens de afstudeerperiode.

Roel Konieczny

Nijmegen, 7 mei 2004

Inhoudsopgave

Samenvatting

Dit onderzoek richt zich op ICT complexiteit binnen organisaties. Het belangrijkste doel van dit onderzoek is: *Hoe kan complexiteit inzichtelijk worden gemaakt en worden gereduceerd, binnen organisaties, door gebruik te maken van enterprise architectuur?*

Om deze vraag te kunnen beantwoorden is er een literatuurstudie uitgevoerd naar de belangrijkste begrippen, namelijk: complexiteit, organisatie en enterprise architectuur en de onderlinge relaties. Om het onderzoeksdoel in kaart te brengen en formeel vast te leggen is er een methodologie opgesteld die dit mogelijk maakt. De methodologie behandelt de technieken Object Role Modeling (ORM) en systeemontwerp middels logisch geformuleerde dependency-structuren in relatie tot enterprise architectuur en complexiteit binnen organisaties.

Binnen dit onderzoek wordt onder enterprise architectuur het volgende verstaan: enterprise architectuur (EA) is het consistente geheel aan principes en modellen dat richting geeft aan ontwerp, realisatie en sloop van processen, organisatorische inrichting, informatievoorziening en technische infrastructuur op zowel het niveau van de business, de informatievoorziening als het technische niveau. Het werken onder architectuur zorgt ervoor dat losse onderdelen in hun samenhang worden ontworpen.

Met enterprise architectuur kunnen verschillende doelen gerealiseerd worden. Een van die doelen is het reduceren van complexiteit. Een Enterprise Architectuur Raamwerk (EAR) draagt bij aan de uitvoering van enterprise architectuur. Het raamwerk kan inzichtelijk maken waar de complexiteit zich bevindt. De complexiteit is in dit onderzoek vastgelegd door gebruik te maken van ORM-modellen en systeemontwerp middels logisch geformuleerde dependency-structuren. Daarnaast kan enterprise architectuur op meta-niveau vastgelegd worden middels ORM. De ORM-modellen maken de relaties tussen de verschillende views inzichtelijk.

Complexiteit bestaat uit twee kerndelen, namelijk: het aantal distincties (verschijningen van een object) en het aantal connecties (verbindingen tussen de objecten). Als er sprake is van veel distincties en veel connecties wordt er gesproken over complexiteit. Complexiteit is tot op heden niet op een eenduidige manier meetbaar, de voornaamste reden hiervoor is dat ieder verschijnsel anders is en niet zomaar te vergelijken valt met een ander verschijnsel. De lengte van de kortste beschrijving van een systeem is tot nu toe de meest gebruikte vorm om complexiteit te meten. Het idee hierachter is: hoe complexer het systeem is hoe lastiger en omvangrijker het is om in zijn geheel te beschrijven. Hierbij speelt het aspect informatie een belangrijke rol. Als er niet bekend is wat het systeem doet: er is onvoldoende informatie over dat systeem; kan er ook niets zinnigs gezegd worden betreffende de complexiteitsgraad.

De statische toestand van complexiteit is echter niet erg interessant. Belangrijker is de verandering in complexiteit: neemt hij toe of af? Complexiteit neemt toe als het aantal distincties of connecties toeneemt en neemt af als het aantal distincties of connecties afneemt.

Complexiteit is vaak ontstaan uit het verleden, ICT is intussen sterk gegroeid en groeit nog steeds zonder dat er daadwerkelijk zicht op is waar het voor dient. De afstemming tussen de business en de ICT speelde in het verleden nog geen belangrijke rol. De verwachting is dat in de toekomst de ICT nog harder en verder zal groeien.

Complexiteit binnen organisaties heeft een aantal negatieve gevolgen: 79% van de bedrijven geeft aan dat complexiteit de oorzaak is voor het niet bereiken van ICT strategische doelstellingen. Meer dan de helft van die bedrijven is het daadwerkelijk niet gelukt om de ICT doelstellingen te realiseren. Een ander belangrijk gevolg van de ICT complexiteit is de time-to-market. Organisaties zijn niet meer in staat snel te reageren op marktveranderingen omdat de ICT te complex is geworden. Daarnaast nemen door de complexiteit de onderhouds- en beheerkosten sterk toe. Kleine functionele wijzigingen in programma's hebben enorme implementatiekosten tot gevolg vanwege de complexiteit. Als laatste komt de continuïteit van de organisatie zelf in gevaar.

Aan de hand van complexiteitsvragen kan worden bekeken of er een kans bestaat dat een organisatie te maken heeft met ICT complexiteit. Het enterprise architectuur raamwerk maakt de complexiteit inzichtelijk. Per view moet naar voren komen uit hoeveel voor de view relevante distincties en connecties deze bestaat. De ORM-modellen en het systeemontwerp middels logisch geformuleerde dependency-structuren leggen formeel vast welke aspecten ter zake doen.

Verschillende type organisaties hebben te maken met verschillende omvangen van complexiteit. Per type organisatie kan globaal worden vastgelegd welke complexiteitsaspecten een rol spelen. Binnen dit onderzoek is gekozen voor de typering van Mintzberg en is per type organisatie vastgelegd wat de mate van complexiteit is.

Om complexiteit uiteindelijk te reduceren moet er invloed worden uitgeoefend op de complexiteit. Binnen dit onderzoek is hiervoor gebruik gemaakt van het besturingsparadigma van de Leeuw.

Complexiteit (bestuurd systeem) kan beïnvloed worden door een besturend orgaan. In het onderzoek is de architect die gebruik maakt van enterprise architectuur het besturend orgaan. Om de complexiteit te verminderen is het zaak per view het aantal connecties en distincties te verminderen. De omgeving waar een bijdrage aan geleverd wordt is de organisatie. Na het uitvoeren van projecten die complexiteit reduceren ontstaat er een organisatie met minder complexiteit.

1 Inleiding

Binnen veel organisaties is behoefte aan een betere en een beter beheersbare ICT inrichting. ICT blijft de laatste decennia maar groeien. Hadden organisaties in het verleden nog redelijk zicht wat er zich op ICT gebied binnen de organisatie afspeelde, nu is er een tijd aangebroken dat organisaties niet meer precies weten wat ze aan boord hebben. Voor veel organisaties is ICT een zeer complexe aangelegenheid geworden. Er zijn bijvoorbeeld systemen en applicaties die min of meer hetzelfde werk verrichten, maar toch allemaal in leven worden gehouden, omdat men niet weet wat de gevolgen zijn als er een systeem of applicatie weggehaald wordt. De vraag is: hoe zou dit beter kunnen?

Dit onderzoek richt zich op het verschijnsel “complexiteit van ICT binnen organisaties”. Er zal onderzocht worden wat complexiteit is, waar het zich bevindt, wat de gevolgen zijn, hoe het ontstaan is en gekeken worden hoe complexiteit gereduceerd kan worden. Enterprise architectuur zal dienen als het middel om complexiteit inzichtelijk te maken en te reduceren. Er zal onderzocht worden wat er onder enterprise architectuur verstaan wordt en wat het nut van enterprise architectuur is.

Over het onderzoeksgebied zal literatuur worden verzameld om de opgestelde hoofdvraag en deelvragen te beantwoorden (zie paragraaf 1.1). Deze literatuur dient als input voor de Object Role Modeling (ORM) modellen en het systeemontwerp middels logisch geformuleerde dependency-structuren. Aan de hand van de literatuur en de ontwikkelde methodologie (zie appendix D) zal het onderzoeksgebied gevisualiseerd worden middels ORM-modellen. Het doel van deze visuele modellen is om inzicht te verschaffen in de relaties en de beïnvloeding binnen het probleemgebied (zie paragraaf 1.2). Daarnaast wordt kenbaar gemaakt wat de relaties zijn tussen de onderlinge gebieden.

Het probleemgebied zal tevens door middel van systeemontwerp middels logisch geformuleerde dependency-structuren in kaart worden gebracht. Dit heeft als doel het onderzoeksgebied als systeem te omschrijven.

Er zal daarnaast een case worden opgesteld om te toetsen of de opgestelde ORM-modellen werkbaar zijn (“proof of concept”).

Als laatste zal er een methodologie worden opgesteld dat ter verantwoording dient van dit onderzoek. Enterprise architectuur is tot op heden nog niet in kaart gebracht middels ORM en systeemontwerp middels logisch geformuleerde dependency-structuren. Om dit te kunnen doen is er een aangepaste methodiek nodig. ORM en systeemontwerp middels logisch geformuleerde dependency-structuren dienen onder de loep te worden genomen om te bekijken welke onderdelen van nut zijn voor het in kaart brengen en visualiseren van het onderzoeksgebied. Daarnaast dient een stappenplan opgesteld te worden dat inzicht verschaft in hoe het probleemgebied vertaald moet worden richting ORM-modellen en systeemontwerp middels logisch geformuleerde dependency-structuren. De methodologie dient ter verantwoording voor de invulling van de ORM-modellen en het systeemontwerp middels logisch geformuleerde dependency-structuren.

1.1 Doel en vraagstelling van het onderzoek

Zoals in de inleiding omschreven is, hebben veel organisaties last van ICT complexiteit. De vragen van dit onderzoek zijn daarom als volgt:

Hoe kan complexiteit door middel van enterprise architectuur in kaart worden gebracht?
Wat zijn de mogelijkheden om complexiteit te kunnen beïnvloeden / reduceren door gebruik te maken van enterprise architectuur met in het bijzonder gebruik te maken van een enterprise architectuur raamwerk (EAR)?

Het onderzoek richt zich met name op de theorie. Aan de hand daarvan zullen de hoofd- en deelvragen beantwoord worden. Binnen dit onderzoek staat één vraag centraal, namelijk de hoofdvraag:
Hoe kan complexiteit inzichtelijk worden gemaakt en worden gereduceerd, binnen organisaties, door gebruik te maken van enterprise architectuur?

Om deze hoofdvraag te kunnen beantwoorden zullen er een aantal deelvragen worden opgesteld die een bijdrage leveren aan het beantwoorden van de hoofdvraag.
De deelvragen luiden als volgt:

- Wat is enterprise architectuur?
- Wat is het nut van enterprise architectuur?
- Wat is een organisatie?
- Welke type organisaties vanuit ICT oogpunt zijn er?
- Wat is complexiteit?
- Hoe is complexiteit af te lezen?
- Is complexiteit meetbaar?
- Hoe is ICT complexiteit ontstaan?
- Wat zijn de gevolgen van ICT complexiteit?
- Hoe kan complexiteit gereduceerd worden?
- Hoe kan complexiteit in kaart worden gebracht met enterprise architectuur?
- Welke relatie bestaat er tussen complexiteit en organisatie?

1.2 Probleemgebied

Om antwoord te geven op de vragen is het nodig een probleemgebied te definiëren. In het probleemgebied worden de belangrijkste onderwerpen kort besproken en de relaties tussen de onderwerpen aangegeven.

Veel organisaties beschikken over uitgebreide ICT voorzieningen om de primaire en secundaire bedrijfsprocessen te ondersteunen. Er is bij bedrijven niet precies duidelijk welke ICT voorzieningen nu wel of niet noodzakelijk zijn en hoe de onderlinge relaties zijn. Dit is per type bedrijf verschillend. Verschillende type bedrijven hebben verschillende ICT eisen. Daarom wordt een organisatie als een probleemgebied gezien.

Vanwege het feit dat er bij bedrijven geen duidelijk overzicht is welke verschillende soorten ICT voorzieningen er zijn en niet bekend is wat het aantal ICT systemen is en welke relaties de ICT systemen onderling hebben, kan er gesproken worden over complexiteit. Complexiteit is ook een probleemgebied op zich.

ICT kan in kaart worden gebracht met enterprise architectuur door gebruik te maken van een architectuurraamwerk. Met behulp van een raamwerk kan er invloed worden uitgeoefend op de ICT binnen de organisatie. Enterprise architectuur wordt daarom gezien als het middel om de probleemgebieden in kaart te brengen / te beïnvloeden.

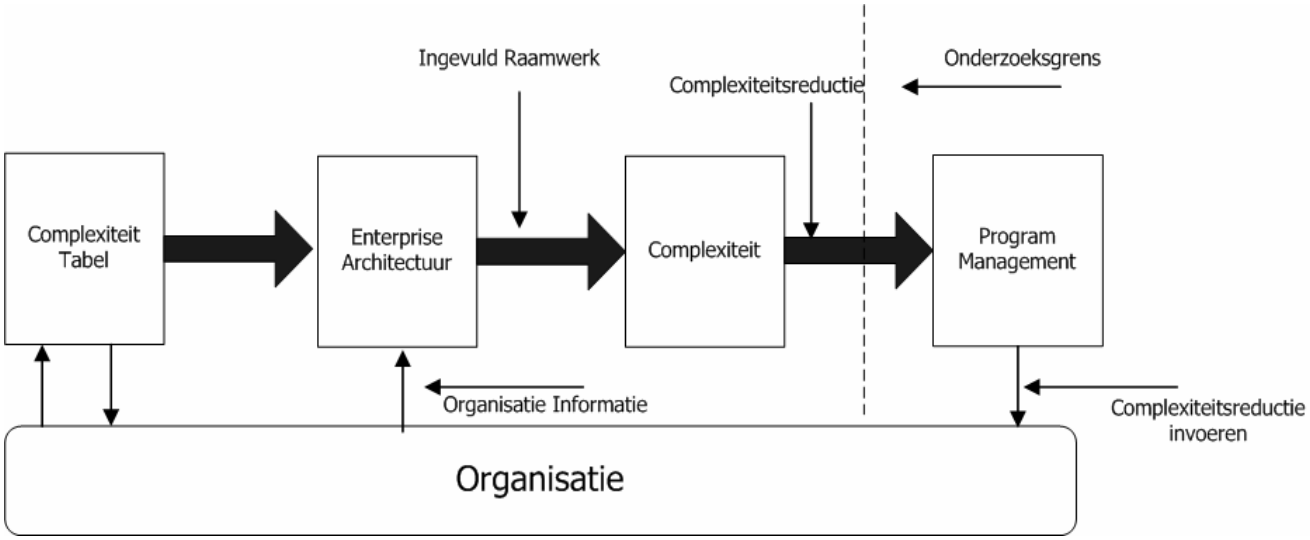
De probleemgebieden binnen dit onderzoek zijn:

- Organisatie;
- Complexiteit.

Het middel binnen dit onderzoek is:

- Enterprise Architectuur.

Figuur 1 geeft de relaties tussen de probleemgebieden en het middel weer.



Figuur 1 Het probleemgebied

De complexiteittabel (zie paragraaf 3.5) is een tabel met een aantal vragen. Deze vragen kunnen gesteld worden aan een organisatie om te toetsen of er wellicht complexiteit aanwezig is. Als er geconstateerd wordt dat er wellicht sprake is van complexiteit kan er worden overgegaan tot het opstellen van een EAR. Dit is de uitkomst van de enterprise architectuur. Aan de hand van het ingevulde raamwerk kan bepaald worden waar de complexiteit zich bevindt. Als er bekend is waar de complexiteit zich bevindt kan er worden overgegaan tot het opstellen van projecten om complexiteit te reduceren (zie paragraaf 4.1). Deze opgestelde projecten dienen vervolgens te worden uitgevoerd. Dit gebeurt middels Program Management. Hier bevindt zich tevens de grens van dit onderzoek. Na het uitvoeren van de projecten ontstaat er een organisatie met minder complexiteit.

1.3 Opbouw Onderzoek

In paragraaf 1.1 is aangegeven dat dit onderzoek met name theoretisch zal zijn. Er wordt onderzocht wat de relaties tussen de probleemgebieden zijn en hoe ze elkaar kunnen beïnvloeden. Het onderzoek heeft een exploratief karakter, omdat er een verkenning plaatsvindt binnen het onderzoeksgebied. Op het moment bestaat er nog geen inzicht in het probleemgebied en zal er aan de hand van de beschikbare literatuur moeten worden onderzocht hoe het probleemgebied in elkaar steekt.

Vanuit de theorie zal gekeken worden naar de verschillende deelvragen. Deze deelvragen zullen in de hoofdstukken 2 tot en met 4 besproken worden. Aan de hand van de deelvragen zal de hoofdvraag beantwoord worden. Met behulp van ORM zullen er modellen worden opgesteld, die inzichtelijk maken waar complexiteit zich bevindt.
Er zal een case worden opgesteld om te toetsen of de opgestelde ORM-modellen en het systeemontwerp middels logisch geformuleerde dependency-structuren werkbaar zijn (“proof of concept”).

1.4 Relevantie

Dit onderzoek is met name relevant voor de middelgrote en grote bedrijven die gebruik maken van veel ICT en van veel verschillende soorten ICT en geïnteresseerd zijn complexiteit te willen reduceren. Daarnaast heeft dit onderzoek een algemeen karakter en is het relevant voor personen die inzicht willen krijgen in de relaties tussen Organisatie, Enterprise Architectuur en Complexiteit.

2 Enterprise Architectuur

Allereerst zal het middel enterprise architectuur worden behandeld: wat is enterprise architectuur eigenlijk? Er zijn tal van verschillende omschrijvingen van het begrip enterprise architectuur te vinden. De meest bekende / gebruikte definitie is die van het Institute of Electrical and Electronics Engineers (IEEE), namelijk de IEEE 1471-2000 Recommended Practice for Architectural Description of Software-Intensive Systems.

Een belangrijk aspect van enterprise architectuur is dat het geen doel op zich is en kan zijn, maar een middel is waarmee verschillende doelen gediend kunnen worden. Deze doelen variëren van het verschaffen van inzicht en overzicht tot en met het plannen, aansturen, monitoren en bijsturen van organisaties.

2.1 Architectuur Definities

Op het moment is er nog geen algemeen geaccepteerde definitie van enterprise architectuur. Hieronder zullen de meest gebruikte definities behandeld worden.

> *The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.*

IEEE 1471 [IEEE]

Deze definitie richt zich met name op de constructie en de relaties van een systeem. Er wordt daarnaast ook rekening gehouden met de omgeving en de principes. Het grootste gemis is naar mijn mening het aspect van de gebruikers / beleving en de relatie met de organisatie. Deze definitie is daarom ook het beste bruikbaar voor software-engineers.

De Gartner Group, hanteert de volgende definitie van enterprise architectuur.

> *IT architecture is a series of principles, guidelines or rules used by an enterprise to direct the process of acquiring, building, modifying and interfacing IT resources throughout the enterprise. These resources can include equipment, software, communications, development methodologies, modeling tools and organizational structures.*

Gartner Group [Gartner]

De Gartner Group richt zich meer op de organisatie en de bedrijfsprocessen. Zij geven aan dat het een middel is om de processen van een bedrijf af te stemmen met de IT middelen. IT middelen kunnen bestaan uit: software, communicatiemiddelen, ontwikkel methodes, modellering gereedschappen en organisatiestructuren. Deze definitie is naar mijn mening breder inzetbaar en houdt ook meer rekening met het bedrijfskundige aspect.

Cap Gemini Ernst & Young (CGEY) heeft als ICT dienstverlener een eigen visie op enterprise architectuur. Zij verrichten architectuuropdrachten voor derden, en hebben een eigen architectuurmethode ontwikkeld. De definitie van Cap Gemini Ernst & Young luidt als volgt:

> Architectuur bestaat uit een coherente en consistente verzameling principes, verbijzonderd in regels, richtlijnen en standaards – soms vastgelegd in ‘patterns’ – die beschrijft hoe een onderneming, de informatievoorziening, een informatiesysteem of een infrastructuur is vormgegeven en zich voordoet in het gebruik.

Cap Gemini Ernst & Young [RIJS]

Cap Gemini Ernst & Young hecht veel waarde aan de principes, regels en richtlijnen. Zij voeren dit nog een stap verder door en maken ook gebruik van standaards en patterns. Daarnaast wordt er rekening gehouden met de organisatie, de informatievoorziening, de informatiesystemen en infrastructuren.

Sogeti is net als Cap Gemini Ernst & Young een ICT dienstverlener. Sogeti heeft een eigen visie over enterprise architectuur. Deze visie luidt als volgt:

> *Architectuur is het consistente geheel aan principes en modellen dat richting geeft aan ontwerp en realisatie van processen, organisatorische inrichting, informatievoorziening en technische infrastructuur van een organisatie. Het werken onder architectuur zorgt ervoor dat losse onderdelen in hun samenhang worden ontworpen, zowel op het niveau van de business, de informatievoorziening als de technische middelen.*

Sogeti [SOG]

De definitie van Sogeti houdt rekening met een breed aantal aspecten van enterprise architectuur, zowel organisatorisch als technisch. Naar mijn mening omvat deze definitie de belangrijkste aspecten van enterprise architectuur.

2.2 Overeenkomsten en verschillen

De grootste overeenkomst tussen de definities is dat ze allemaal aandacht besteden aan de onderliggende principes van het te ontwikkelen of bestaande systeem. Daaruit volgen een aantal regels met bijbehorende richtlijnen. Het aspect constructie en de onderlinge relaties komt ook bij alle vier de definities naar voren.

Het grootste verschil tussen de definities is de afstemming tussen de organisatie en de ICT. Met name de definitie van IEEE houdt geen rekening met de bedrijfsprocessen en de afstemming. IEEE richt zich voornamelijk op constructie.
De definitie van Sogeti houdt met het grootst aantal aspecten van enterprise architectuur rekening.

Al deze definities houden niet expliciet rekening met het reduceren van complexiteit. Tijdens het landelijk architectuur congres 2003 is er een definitie opgesteld die rekening houdt met complexiteit. Deze definitie luidt als volgt:

> Architectuur bestaat uit een consistente set van principes voor ontwerp, realisatie en sloop op zowel het niveau van een organisatie (enterprise architectuur) als op het niveau van een systeem (software architectuur).

[BERG]

Het aspect “sloop” in deze definitie omvat de aanpak om bepaalde ICT systemen uit een organisatie te “slopen” oftewel het reduceren van de ICT en daarmee een minder complexe situatie te krijgen. Om een werkbare definitie voor dit onderzoek te krijgen zijn de definities van Sogeti en de definitie van van de Berg met elkaar gecombineerd.

> Architectuur is het consistente geheel aan principes en modellen dat richting geeft aan ontwerp, realisatie en sloop van processen, organisatorische inrichting, informatievoorziening en technische infrastructuur op zowel het niveau van de business, de informatievoorziening als het technische niveau. Het werken onder architectuur zorgt ervoor dat losse onderdelen in hun samenhang worden ontworpen.

Binnen dit onderzoek zal de definitie zoals hierboven opgesteld is gebruikt worden, omdat deze rekening houdt met de twee andere probleemgebieden: complexiteit (d.m.v. het sloop aspect) en organisatie.

2.3 Nut van architectuur

Nu bepaald is wat onder architectuur verstaan wordt is het belangrijk om te onderzoeken wat het nut van architectuur is. Waarom wordt architectuur toegepast en wat zijn de voordelen van architectuur?

Belangrijke punten waar architectuur in kan bijdragen volgens de onderzoeksgroep Meta group:

- Complexiteitsreductie “complexiteit te managen en vast te leggen, zodat de verschillende belanghebbenden hierover kunnen praten”;
- Kostenbesparing;
- Bevordert intergratie tussen verschillende systemen (intern,extern);
- Verkort ICT ontwikkeltijd;
- Checklist om zeker te zijn dat geen stappen worden overgeslagen;
- Richting en stuurmiddel voor toekomst investeringen;
- Business gedreven introductie van nieuwe technologieën;
- Framework van randvoorwaarden waaraan een informatiesysteem of een groep van informatiesystemen moeten voldoen;
- Bevordert afstemming ICT met de business;
- Dwingend kader hoe te werken;
- Bewaken van consistentie;
- Standaardisatie van de informatievoorziening afdwingen;
- Middel om de organisatie voor te bereiden voor ‘snelle’ ongeplande veranderingen.

[META1]

Complexiteitsreductie kan volgens de Meta group behaald worden door het goed toepassen van architectuur.

Belangrijke punten waar architectuur in kan bijdragen volgens IEEE:

- Analyse middel: om te onderzoeken welke architectuurmethode het best geschikt is;
- Business planmiddel: om vanuit een legacy architectuur naar een nieuwe architectuur te komen;
- Communicatiemiddel tussen organisaties die betrokken zijn bij de ontwikkeling, productie en beheer van het systeem;
- Communicatiemiddel tussen de klant en de ontwikkelaar, als onderdeel van contract onderhandelingen;
- Ontwikkel en onderhoudsdocumentatie: inclusief het materiaal voor hergebruik en opleidingen;
- Invoer voor het nagekomen systeem ontwerp en ontwikkel activiteiten;
- Invoer voor analyse tools;
- Operationele en infrastructuur ondersteuning;
- Plan en budget ondersteuning;
- Review,analyse en evaluatie middel van het systeem door de gehele levensduur;
- Specificatie voor een groep systemen die een gemeenschappelijke bron van functionaliteiten bevatten.

[IEEE]

De Meta group richt zich veel meer op organisaties (enterprise architectuur), terwijl IEEE zich meer richt op systeemontwikkeling (software architectuur).
Er zijn ook een aantal overeenkomsten te onderkennen namelijk, het zien van architectuur als communicatiemiddel, het kostenaspect en het planmiddel.

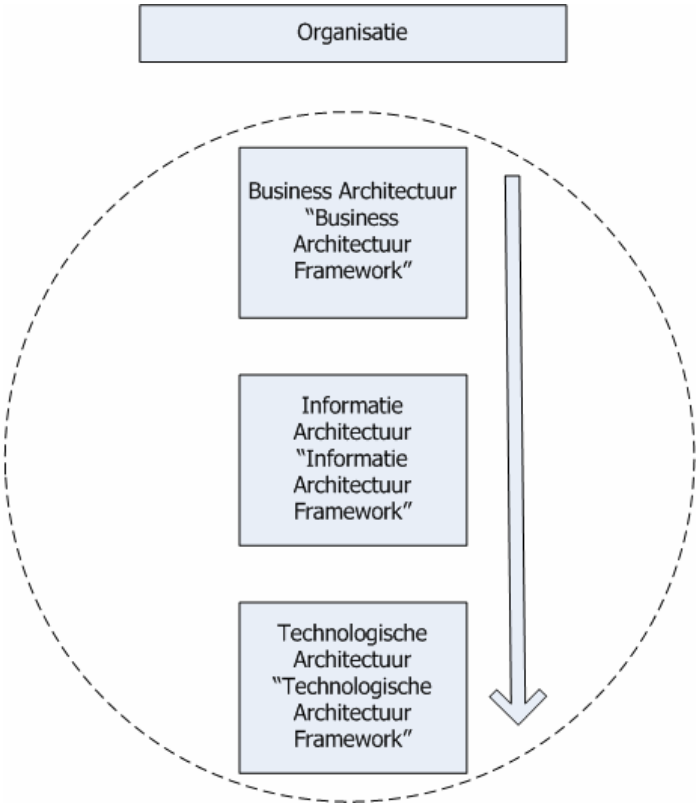
Binnen dit onderzoek wordt architectuur als enterprise architectuur gezien, omdat het een architectuur is die rekening houdt met de organisatie en niet alleen met het ontwikkelen van systemen.
Binnen de enterprise architectuur van een organisatie bestaan verschillende omvangen van complexiteit. Voordat er verder wordt ingegaan op complexiteit en organisaties wordt eerst enterprise architectuur behandeld.

2.4 Enterprise Architectuur

Enterprise architectuur heeft drie belangrijke gebieden. Deze drie gebieden luiden als volgt:

- Business architectuur;
- Informatie architectuur;
- Technologische architectuur.

[META1]



Figuur 2 Enterprise Architectuur

Per architectuurgebied worden de doelen bepaald die gerealiseerd moeten worden en wordt bepaald welke producten daar uit voortvloeien. De producten op een hoger niveau binnen de enterprise architectuur vormen steeds de uitgangssituatie voor de producten op het daaronder liggende niveau. Dit kan goed worden vergeleken met de watervalaanpak. Het grote verschil is dat de bovenliggende gebieden op voorhand al rekening moeten houden wat er op de onderliggende gebieden aanwezig is. Enterprise architectuur werkt als volgt:
de organisatie bepaalt de business architectuur, voortvloeiend uit de strategie, de business architectuur vormt de input voor de informatie architectuur en de informatie architectuur vormt de input voor de technologische architectuur, rekeninghoudend met elkaar. De business wordt als leidend gezien.

2.4.1 Business Architectuur

De business architectuur geeft een beeld van de eisen die de business op korte en lange termijn stelt aan de organisatie en de informatievoorziening. Daarnaast behandelt de business architectuur de keuzes die gemaakt zijn met betrekking tot de organisatie-inrichting en de informatievoorziening om aan deze eisen te kunnen voldoen. Centraal staat hier de visie van het management op de (door de ICT) veranderende markt en op de noodzakelijke veranderingen in de bedrijfsvoering. Deze visie vormt de input voor de Informatie architectuur.

2.4.2 Informatie architectuur

De informatie architectuur geeft een overzicht van de eisen die aan het totale informatiesysteem worden gesteld. Het geheel wordt opgesplitst in deelsystemen zodat aan deze eisen kan worden voldaan. Binnen deze deelsystemen worden de functionaliteiten beschreven en de onderlinge samenhang. Het eindproduct vormt enerzijds de basis voor de ontwikkeling van de software-architectuur. Anderzijds is het nodig voor de ontwikkeling van het migratieplan.

Het Genootschap van Informatie Architecten (GIA) is al vele jaren actief als beroepsvereniging voor professionals op het gebied van architectuur. Zij hebben een definitie van informatie architectuur opgesteld.

Definitie Genootschap voor Informatie Architecten [GIA]

Een informatie architectuur is een samenhangende visie van een organisatie op haar bestaande en gewenste informatievoorziening. Een informatie architectuur komt tot stand door een gezamenlijk proces van beeldvorming van en onderhandeling tussen alle betrokkenen. In een informatie architectuur komen de elementen van de informatievoorziening en hun samenhang tot uitdrukking, alsmede hun aansluiting op de bedrijfsarchitectuur en de ICT-architectuur en het waarom hiervan. Inherent aan een informatie architectuur zijn keuzes op het gebied van informatiefunctionaliteiten en –structuren. Deze keuzes worden vastgelegd in de vorm van principes, standaarden en modellen. Daarmee is een informatiearchitectuur het bestemmingsplan voor de vernieuwing van de informatievoorziening van een organisatie.

De definitie van GIA is vrij breed en houdt rekening met de bestaande informatievoorziening en de communicatie met de direct betrokkenen. Dit aspect komt in de praktijk vaak voor. Verder belichten zij het aspect tussen het bedrijfsgebeuren en de ICT en wordt kenbaar gemaakt dat binnen de informatiearchitectuur keuzes gemaakt dienen te worden betreffende de informatiefunctionaliteiten en –structuren.

2.4.3 Technologische architectuur

De technologische architectuur moet een beeld geven van de eisen die aan de software worden gesteld en beschrijft de technische oplossingen die zijn gekozen om aan die eisen te voldoen. Concrete regels en richtlijnen beschrijven hoe de software tijdens de bouw moet worden gestructureerd.

Er wordt vastgelegd hoe conceptuele componenten (uit de informatie architectuur) verder opgesplitst moeten worden tot softwarecomponenten, hoe deze softwarecomponenten onderling communiceren en hoe ze over de technische infrastructuur moeten worden verdeeld.
[ISES]

2.5 Enterprise Architectuurraamwerk

Een EAR is een hulpmiddel om op gestructureerde wijze je enterprise architectuur in kaart te brengen, om bepaalde doelen te bereiken (zie paragraaf 2.3), in dit geval dus het verminderen van complexiteit.

Een EAR bestaat uit een aantal gezichtspunten (viewpoints) op een organisatie. Dat wil zeggen vanuit verschillende invalshoeken wordt de organisatie onder de loep genomen. De gezichtspunten kan je onderverdelen in 3 hoofdgebieden (zie paragraaf 2.4) namelijk:

- Business Architectuur;
- Informatie Architectuur;
- Technologische Architectuur.

Binnen deze drie gebieden worden een aantal stappen uitgevoerd om uiteindelijk tot een architectuur oplossing te komen. Het EAR moet er voor zorgen dat de drie onderdelen op elkaar zijn afgestemd, om zodoende een Business-ICT alignement te krijgen. Ieder raamwerk heeft hiervoor zijn eigen aanpak en gebruikt verschillende soorten terminologie, terwijl de betekenis min of meer hetzelfde is. Binnen dit onderzoek zal de definitie zoals in paragraaf 2.1 opgesteld is worden gebruikt. Deze definitie houdt namelijk rekening met het aspect “sloop” en een breed aantal aspecten van enterprise architectuur. Het Raamwerk dat daarbij behoort is in Figuur 3 weergegeven.

	Businessdoelen							
	Business-architectuur			Informatie-architectuur		Technische architectuur		
	Prod/ dienst	Proces	Orga-nisatie	Gege-vens	Appli-catie	Middle-ware	Plat-form	Net-werk
Algemene principes								
Beleidslijnen								
Modellen								

Figuur 3 Sogeti Raamwerk [SOG]

Omdat een raamwerk naar alle ICT aspecten kijkt binnen een organisatie is het niet relevant om al deze aspecten te behandelen. De focus zal liggen op de aspecten die de complexiteit van de ICT kunnen beïnvloeden.

De horizontal as behandelt de Algemene principes; een principe behandelt het bestaansrecht van een domein. Dit wordt per domein (die op de verticale as staan), behandeld. De domeinen zijn middels ORM gemodelleerd en door systeemontwerp middels logisch geformuleerde dependency-structuren omschreven. De tabellen die bij de domeinen horen zijn in paragraaf 3.5 beschreven.

De Beleidslijnen behandelen de regels en richtlijnen die per domein voorkomen. Een regel moet uitgevoerd worden. Een richtlijn hoeft niet perse uitgevoerd te worden, maar het zou mooi zijn als daar aan voldaan werd.

De modellen geven een visuele representatie van de domeinen, verschaffen inzicht, omschrijven de relaties en houden rekening met de principes,regels en richtlijnen. Tezamen vormen zij een consistent geheel. Figuur 13, Figuur 14 en Figuur 15 laten middels ORM zien hoe architectuur opgebouwd is, en houden rekening met het raamwerk van Figuur 3. De view organisatie komt niet voor in het ORM-model, in paragraaf 3.6 zal deze view besproken worden.

2.6 Samenvatting

Binnen dit onderzoek wordt onder enterprise architectuur het volgende verstaan: enterprise architectuur (EA) is het consistente geheel aan principes en modellen dat richting geeft aan ontwerp, realisatie en sloop van processen, organisatorische inrichting, informatievoorziening en technische infrastructuur op zowel het niveau van de business, de informatievoorziening als het technische niveau. Het werken onder architectuur zorgt ervoor dat losse onderdelen in hun samenhang worden ontworpen.

Met enterprise architectuur kunnen verschillende doelen gerealiseerd worden. Een van die doelen is het reduceren van complexiteit. Een Enterprise Architectuur Raamwerk (EAR) draagt bij aan de uitvoering van enterprise architectuur. Het raamwerk kan inzichtelijk maken waar de complexiteit zich bevindt. De complexiteit is in dit onderzoek vastgelegd door gebruik te maken van ORM-modellen en systeemontwerp middels logisch geformuleerde dependency-structuren. Daarnaast kan enterprise architectuur op meta-niveau vastgelegd worden middels ORM. De ORM-modellen maken de relaties tussen de verschillende views inzichtelijk.

3 ICT Complexiteit

Voordat er dieper op het ontstaan van complexiteit wordt ingegaan wordt er eerst besproken wat complexiteit is. Complexiteit zal niet behandeld worden vanuit de wiskundige / informatica kant. De voornaamste reden hiervoor is dat enterprise architectuur nog niet wiskundig te omschrijven valt. Daarnaast zal een wiskundige aanpak “te zwaar” zijn voor dit onderzoeksgebied, een wiskundige benadering is ook niet noodzakelijk om ICT complexiteit binnen organisaties boven tafel te krijgen.

Er wordt in dit onderzoek onderzocht of complexiteit meetbaar is en of het te constateren valt. Vervolgens worden de gevolgen van complexiteit besproken en wordt complexiteit in relatie met enterprise architectuur gebracht. De viewpoints uit het EAR zullen behandeld worden. Als laatste zal er gekeken worden naar de organisatie in relatie tot complexiteit.

3.1 Complexiteit in zijn algemeenheid

Complexiteit is oorspronkelijk een Latijns woord (*complexus)* en betekent samengevouwen. Dit houdt in dat iets complex is als het twee of meer delen heeft die zodanig met elkaar verstrengeld zijn dat ze moeilijk uit elkaar zijn te halen. Hieruit kan worden afgeleid dat complexiteit bestaat uit distincties (onderdelen) en connecties (verbindingen), die onderling een relatie hebben.

Distincties zijn het aantal verschijningen (onderscheidingen) van een object. Bijvoorbeeld het aantal verschijningen van het object besturingssysteem (Windows XP,2000,98), hier komen dus drie verschillende distincties van het object besturingsysteem voor. Bij iedere distinctie behoort een aantal, bijvoorbeeld Windows XP (10x), Windows 2000 (16x) en Windows 98 (100x).

Connecties zijn de verbindingen tussen de distincties. Een distinctie kan ook een connectie hebben met dezelfde distinctie. Bijvoorbeeld Windows XP (1) heeft een relatie met Windows XP (5), dit geldt als 1 connectie. Een ander voorbeeld van een connectie is het volgende: Windows XP (1) heeft een relatie met Windows 98 (78); dit geldt ook als 1 connectie.

Complex heeft ook vaak de betekenis “moeilijk” gekregen. Dit komt omdat systemen met veel gerelateerde componenten moeilijk te analyseren zijn en dus ook moeilijk zijn te begrijpen. Naar mijn mening is complexiteit toch iets anders dan moeilijk. Iets complex kan intern uit makkelijk te begrijpen onderdelen bestaan, en is dus niet moeilijk op te lossen. Als iets complex is dan is er geen duidelijk overzicht / inzicht meer in de distincties en de connecties, maar dit zegt niks over de moeilijkheidsgraad. Vaak is iets dat complex is “moeilijk”, maar dit geldt niet altijd.

Er zijn twee belangrijke begrippen die bij de discussie over complexiteit naar voren komen namelijk:

- Orde;
- Wanorde.

Intuïtief wordt orde vaak omschreven als regulariteit of regelmaat oftewel het volgen van vastgestelde regels. Binnen complexiteit wordt er onder orde het volgende verstaan:

> Orde wordt gekenmerkt door veel connecties, maar weinig distincties.

In een geordend systeem zijn de distincties homogeen of repetitief van aard. Als voorbeeld wordt besturingsysteem genomen: Een organisatie heeft alleen maar Windows 98 besturingssystemen die onderling verbonden zijn bijvoorbeeld: 100 Windows 98 systemen zijn allemaal met elkaar verbonden. De configuratie van één systeem is voldoende om de rest te weten.

Het tegenovergestelde van orde is wanorde. Onder wanorde kan het volgende worden verstaan:

> Wanorde wordt gekenmerkt door veel verschillende distincties, maar weinig connecties.

In een wanordelijk systeem zijn de distincties verschillend en onafhankelijk van elkaar. Als voorbeeld wordt besturingsysteem genomen: Een organisatie heeft Windows 98, 95, XP, 2000, ME, Linux 7.0, Linux 6.0. Sommige besturingssystemen zijn onderling verbonden en andere niet. Met de configuratie van één systeem kan er nog niks gezegd worden.

Complexiteit zit tussen orde en wanorde in, zie Tabel 1 complexiteit

	Distincties	**Connecties**
Orde	Weinig	Veel
Complexiteit	Veel	Veel
Wanorde	Veel	Weinig

Tabel 1 complexiteit

[EDMONDS], [HEYLIGHEN]

3.1.1 Complexiteit meetbaar?

Het zou mooi zijn om te kunnen aangeven hoe complex iets is, bijvoorbeeld “dit is 10 complex op de schaal van 1 tot en met 100”. In het verleden zijn er tientallen pogingen geweest om complexiteit te definiëren op een meetbare, kwantificeerbare manier, zodat er van een willekeurig verschijnsel vastgesteld kan worden hoe complex het is. Geen enkele van die methodes is echter algemeen bruibaar omdat ieder verschijnsel anders is en niet zomaar te vergelijken valt met een ander verschijnsel. Bijvoorbeeld: is een helikopter complexer dan een tank of een vliegtuig? Deze verschijnsels zijn niet echt te vergelijken, en kan er dus ook geen waarde aan worden gegeven. Een motorfiets, anderzijds, is complexer dan een fiets, aangezien een motorfiets alle onderdelen heeft van een fiets plus nog extra onderdelen.

Een manier om complexiteit te kunnen vaststellen is om te kijken naar de omvang van het systeem / verschijnsel. Hoe groter de omvang is hoe complexer het is. Deze methode geldt echter niet in iedere situatie, bijvoorbeeld: volgens deze methode is een doos met spijkers complexer dan een microprocessor. Een microprocessor is veel complexer dan een doos met spijkers omdat in een microprocessor heel veel connecties bestaan en in een doos met spijkers niet.

Een andere manier is het opstellen van algoritmes die bepalen hoe complex iets is. Deze algoritmes werken alleen als het systeem op een formele “wiskundige” manier is omschreven. Enterprise architectuur en ICT complexiteit binnen organisaties zijn niet formeel en wiskundig vastgelegd (zie appendix D). Deze algoritmes kunnen daarom niet gebruikt worden.

De beste definitie tot nu toe om te bepalen hoe complex iets is wat niet wiskundig en formeel is vastgelegd is de lengte van de kortste beschrijving van het systeem, beter bekend als de Kolmogorov methode [CAST], [CORN]. Het idee hierachter is: hoe complexer het systeem is hoe lastiger en omvangrijker het is om in zijn geheel te beschrijven. Hierbij speelt het aspect informatie een belangrijke rol. Als er niet bekend is wat het systeem doet: er is onvoldoende informatie over dat systeem; kan er ook niets zinnigs gezegd worden betreffende de complexiteitsgraad. Er bestaat daarnaast een risico dat er dingen worden weggelaten (omdat er informatie ontbreekt) zodat het systeem minder complex lijkt.
Er is dus informatie nodig om het systeem compleet te kunnen beschrijven. Als er geen informatie aanwezig is wordt er gesproken over entropie.

Entropie is de mate van onzekerheid over een systeem. Dat wil zeggen dat niet precies bekend is wat het systeem daadwerkelijk doet. Over een systeem met een hoge entropie kan weinig gezegd worden, dus ook niet of het wel of niet complex is. Hiervoor is informatie nodig.

Informatie is hetgeen dat gebrek aan kennis of de onzekerheid opheft. Informatie kan een afname van entropie veroorzaken.

De volgende formule illustreert de relatie tussen entropie en informatie

E(na) = E(voor) - I

E (na): de onzekerheid over een systeem nadat er informatie is verkregen.
E (voor): de onzekerheid over een systeem voordat er informatie verkregen.
I: De verkregen informatie over het systeem.

In het meest gunstige geval is de E(na) 0. Dat wil zeggen dat er geen informatie over het systeem ontbreekt. In veel gevallen is dit niet te realiseren, omdat ieder detail van het systeem naar voren moet komen. E (na) moet uiteindelijk een waarde krijgen die voldoende is om de complexiteit van een systeem te kunnen omschrijven. Als er bijvoorbeeld gekeken wordt naar het domein platform uit het EAR (zie paragraaf 2.5) en de entropie over het domein platform is op het moment 40 op de schaal van entropie (E voor) dan kan door middel van het verkrijgen van de juiste informatie (I) de entropie afnemen. Na het analyseren en bestuderen van de documentatie over het platform is er 20 aan Informatie gevonden. De formule kan nu worden ingevuld:

Entropie Platform:
E (na) = E (40) – I (20)
E (na) = 20

Het probleem dat hier een rol speelt is het toekennen van een waarde aan informatie. Omdat er niet altijd bekend is hoeveel informatie er ontbreekt is het erg lastig aan te geven hoeveel informatie er boven tafel is gekomen.

Het abstractie niveau speelt ook een belangrijke rol. Neem bijvoorbeeld de complexiteit van een platform in de enterprise architectuur. Het is daarbij onder andere belangrijk om te weten welk besturingssysteem het platform bevat. Echter is het niet noodzakelijk om alle details van het betreffende besturingssysteem te achterhalen om iets over de complexiteit van het platform te zeggen. Om de complexiteit van een systeem te kunnen omschrijven dient er voldoende informatie aanwezig te zijn over het systeem of dient deze informatie verkregen te worden.

De statische toestand van complexiteit binnen dit onderzoek is eigenlijk niet echt interessant. Het is interessanter om de verandering van complexiteit vast te stellen. Om na te gaan of de complexiteit eerder toeneemt of afneemt.

Complexiteit neemt volgens de definitie toe als het aantal distincties en/of connecties toeneemt. Bijvoorbeeld het aantal distincties van besturingssystemen neemt toe. Naast Windows XP, 2000, 98 komt er ook Windows 95 bij. Hierdoor neemt het overzicht af en wordt het geheel complexer. Het aantal connecties kan ook toenemen, bijvoorbeeld door alle Windows XP machines naast het verbinden met Windows 2000 nu ook te verbinden met Windows 98. Hierdoor neemt het overzicht ook af, en wordt het geheel complexer.

Toename van distincties met afname van connecties betekent toename van wanorde (differentiatie), niet van complexiteit. Bijvoorbeeld naast Windows XP, 2000, 98 komt er ook Windows 95 bij en wordt Windows XP niet meer verbonden met Windows 2000. Dit leidt tot meer wanorde.

Toename van connecties (integratie) met afname van distincties veroorzaakt een toename van orde, bijvoorbeeld door alle Windows XP machines te verbinden met Windows 2000 en Windows 98 geheel te laten verdwijnen.

Complexificatie (complexer worden) kan dus gedefinieerd worden als differentiatie+integratie.
Complexiteit reduceren is hier het tegenovergestelde van, namelijk het afnemen van differentiatie+integratie

[EDMONDS], [HEYLIGHEN]

3.2 Complexiteit te constateren?

Voordat er wordt overgegaan tot het daadwerkelijk constateren van complexiteit binnen een organisatie is het noodzakelijk om te weten welke punten een rol spelen bij complexiteit. Een aantal punten die een rol spelen zijn in Tabel 2 opgenomen.

• Er bestaat geen duidelijk overzicht (totaalplaat) meer van de huidige ICT en de onderlinge relaties tussen de ICT systemen; • Er is niet duidelijk welke invloeden een systeem heeft op andere systemen; • Er is niet duidelijk wat de effecten zijn als dat systeem weggehaald wordt; • Er is onbekend wat de functionaliteit van de ICT systemen zijn, wat doet ieder systeem precies; • De ICT systemen zijn zo oud dat er geen kennis meer aanwezig is bij de huidige ICT medewerkers; • De werkzaamheden worden door de ICT Belemmerd.

Tabel 2 complexiteit constateren

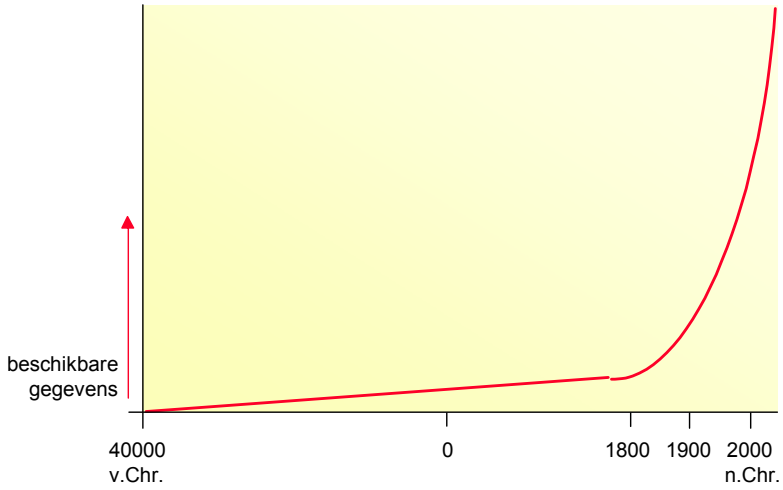
Als aan één van deze bovengenoemde punten is voldaan is de kans aanwezig dat er complexiteit aanwezig is binnen de organisatie. De vraag die hier naar voren komt is: hoe is complexiteit ontstaan? Om dit te kunnen nagaan zal er gekeken moeten worden naar de ICT geschiedenis.

3.3 Het ontstaan van ICT complexiteit

Tot 1970 was ICT alleen weggelegd voor de grootste organisaties en was ICT centraal geregeld.

Het mainframe deed zijn intrede in het bedrijfsleven, en kon alleen simpele rekenintensieve taken uitvoeren, zoals de salarisadministratie. ICT had een beperkte ondersteunende rol in de organisatie en was vrij overzichtelijk.

Tussen 1970 en 1980 brak de ICT echt door in het bedrijfsleven. Naast het mainframe deed de office computer of minicomputer zijn intrede. Kleinere organisaties en bedrijfsafdelingen kregen ook de mogelijkheid om ICT toe te passen. Er ontstond decentralisatie en meerdere applicaties op verschillende platformen. Echter was de omvang nog niet groot. De groei van het aantal gegevens nam ook sterk toe en neemt nog steeds toe zie Figuur 4. [RIJS]



Figuur 4 Toename gegevens [RIJS]

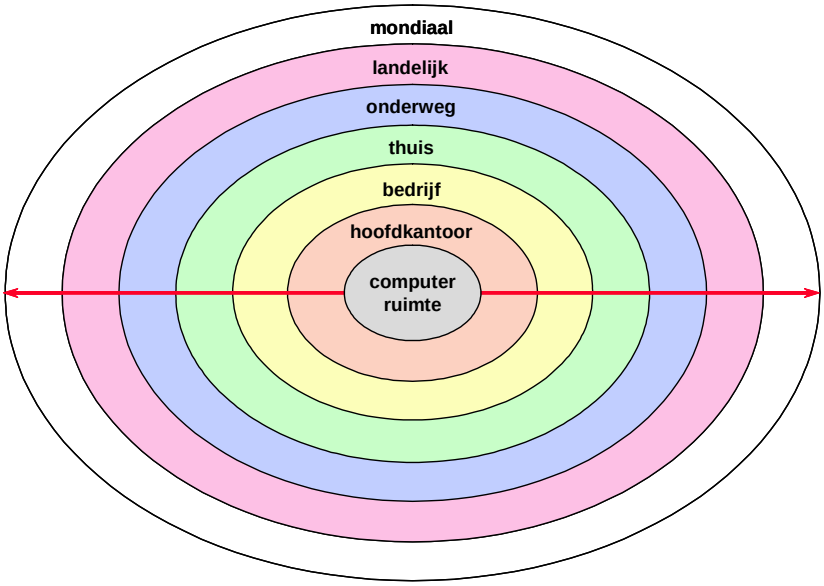
Hoewel de markt voor de ICT tussen 1980 en 1990 nauwelijks veranderde, maar wel flink groeide, werd in deze periode de basis gelegd voor een aantal ontwikkelingen die het daarop volgende decennium zouden beheersen.

Naast het mainframe en de minicomputer deed de personal computer (pc) zijn intrede. Er bestond nu de mogelijkheid om softwarelicenties in pakketvorm te kopen. Voorheen was het alleen mogelijk applicaties aan te schaffen die specifiek voor de klant werden ontwikkeld (maatwerk). De belangrijkste rol van ICT was nog steeds het ondersteunen van secundaire processen, zoals de boekhouding en de administratie. Eind jaren tachtig kwamen de eerste pakketten op de markt die ondersteuning boden aan de primaire processen. De organisatie werd afhankelijk van ICT. Het belang van ICT nam toe en de computersystemen van verschillende organisaties en van verschillende soorten, gingen met elkaar communiceren op basis van datacommunicatie protocollen.

De ICT was sterk technologische gericht en de automatiseringsafdelingen hadden het voor het zeggen. Het overzicht over de ICT ontbrak evenals een totaalplaat. ICT was een ivoren toren, moeilijk grijpbaar, begrijpbaar en nog moeilijker te besturen. Een bedrijfskundige onderbouwing voor het automatiseren was vaak niet aanwezig.

In de jaren '90 raakte de ICT in een stroomversnelling.
De pc werd krachtiger en goedkoper en het Internet deed zijn intrede.

Systemen werden met elkaar verbonden zie Figuur 5. De systemen uit de beginjaren waren vaak nog operationeel. Daar werden interfaces voor gemaakt, zodat de nieuwe systemen met de oude 'legacy' systemen konden blijven communiceren. ICT ging zich veel meer richten op de primaire processen van een organisatie. Er werd meer rekening gehouden met de organisatie (wat men wil), maar nog steeds werden er zaken geautomatiseerd waar de organisatie niet altijd op stond te wachten. De ICT dijde flink uit. Organisaties hebben nu systemen staan uit verschillende tijdperken en geen compleet overzicht meer. ICT is complex geworden.



Figuur 5 Uitbreiding ICT [RIJS]

Recentelijk zijn organisaties kritischer geworden bij het beoordelen van ICT. ICT moet ook daadwerkelijk organisatiedoelen dienen en het moet uiteindelijk geld opleveren.
Het bedrijfsleven en de overheid beperken zich niet meer tot ICT als tactisch instrument, maar nemen ICT ook mee in hun strategie. Enterprise Application Integration (EAI) is de trend van het eerste decennium van de nieuwe eeuw.

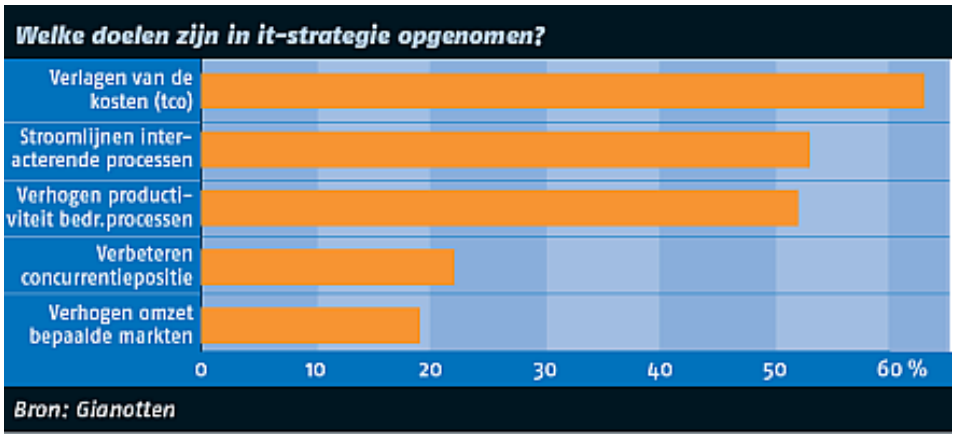
[VAN DER POLS]

3.4 Gevolgen van ICT complexiteit

Als er binnen een organisatie sprake is van complexiteit, wat zijn dan de gevolgen voor die organisatie?
Het onderzoeksbureau Giarte Research heeft een onderzoek gehouden onder chief information officers (CIO) en topmanagers. De conclusie die uit het onderzoek naar voren kwam was als volgt:

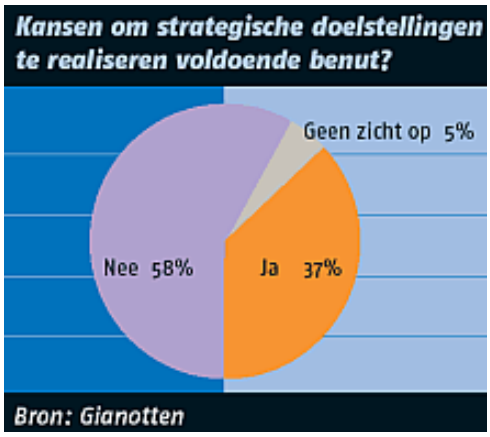
> Ruim 60 procent van middelgrote en grote firma's maakt te weinig gebruik van de kansen die informatietechnologie biedt om strategische doelstellingen te bereiken, zoals het reduceren van de bedrijfskosten en het verhogen van procesintegratie. Maar liefst 79 procent van de geënquêteerden noemt hiervoor de groeiende complexiteit van it als belangrijkste oorzaak.
> [GIANOTTEN]

Figuur 6 toont de doelen die in de IT-strategie zijn opgenomen volgens Giarte Reasearch



Figuur 6 IT-Strategie [GIANOTTEN]

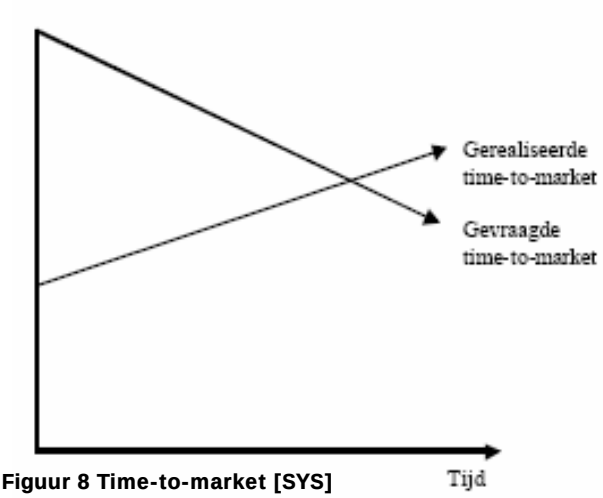
Om deze doelen te kunnen behalen moet de IT-strategie goed uitgevoerd kunnen worden. Maar liefst 79% van de geënquêteerden is minder goed in staat haar IT-strategie uit te voeren vanwege complexiteit. De vraag die dan naar voren komt is: is het die organisaties dan uiteindelijk toch gelukt om haar kansen van de IT-strategie goed uit te voeren ondanks de belemmerende factor van ICT complexiteit? Giarte Research heeft deze vraag binnen haar onderzoek gesteld en de volgende resultaten kwamen naar boven



Figuur 7 Kansen Benut [GIANOTTEN]

Zoals in Figuur 7 te zien is, is meer dan de helft van de organisaties er niet in geslaagd haar strategische doelstellingen voldoende te benutten. Toch is er 37% van de organisaties wel in geslaagd haar ICT kansen te benutten en dat terwijl 21% van de bedrijven ICT complexiteit niet als een belemmerende factor beschouwd. Er zijn dus bedrijven die met de bestaande complexiteit haar IT- strategie voldoende weten te benutten.

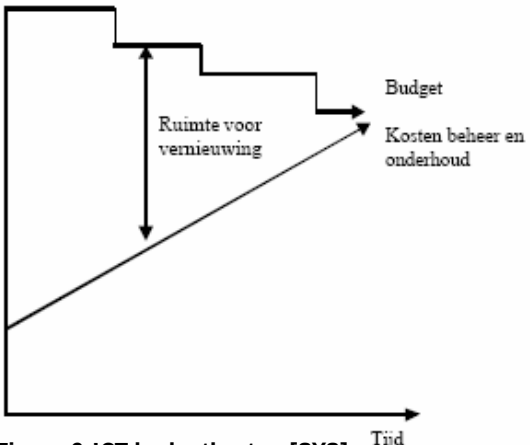
Een gevolg van ICT complexiteit is het niet op tijd kunnen reageren op marktveranderingen. De business eist dat er snel kan worden ingespeeld op veranderende marktaspecten, terwijl de ICT te complex is geworden om snel wijzigingen te kunnen aanbrengen. De time-to-market neemt hierdoor sterk toe zoals Figuur 8 illustreert.



Figuur 8 Time-to-market [SYS]

De gevraagde time-to-market kan niet meer bereikt worden door de ICT, omdat de complexiteit steeds verder toeneemt. Bijvoorbeeld: als een bedrijf een nieuwe dienst wil introduceren en het bedrijf moet daarvoor de bestaande complexe ICT aanpassen is het veel tijd kwijt met het aanpassen van de ICT. De reactie op de markt neemt door die aanpassingen van de ICT af.

Een ander gevolg van de toename van complexiteit is het toenemen van de onderhouds- en beheerkosten. Onderzoek heeft aangetoond dat sommige aanpassingen aan informatiesystemen nodig zijn vanwege het feit dat andere ICT systemen worden aangepast. Het beheerbudget stijgt dus sterk, terwijl er vanuit de organisatie steeds meer wordt bezuinigd op het ICT budget. Figuur 9 illustreert dit gevolg.



Figuur 9 ICT budgetkosten [SYS]

Een derde gevolg van de complexiteit is dat kleine functionele wijzigingen enorme implementatiekosten tot gevolg kunnen hebben.
De toegevoegde waarde van iedere geïnvesteerde Euro neemt daardoor steeds verder af.

Tot slot vormt de complexiteit een risico voor de continuïteit van de onderneming. Consequenties van wijzigingen kunnen vooraf niet worden doorgrond, maar moeten tijdens exploitatie blijken, met alle negatieve consequenties voor de bedrijfsvoering.
Bijvoorbeeld: een organisatie is bezig met het introduceren van een nieuw product, hiervoor dient de ICT te worden aangepast. Omdat deze organisatie te maken heeft met een complexe ICT en geen duidelijk zicht meer heeft op de ICT, is het bedrijf niet in staat om van te voren vast te stellen wat er gewijzigd moet worden. Als het te lang duurt en te kosten te hoog zijn kan de continuïteit van het bedrijf in gevaar komen.
[SYS]

3.5 Complexiteit in relatie met Enterprise Architectuur.

Om ICT complexiteit in kaart te brengen wordt er gebruik gemaakt van enterprise architectuur.

De complexiteit van ICT kan worden onderverdeeld in drie gebieden. Deze drie gebieden komen uit de enterprise architectuur (zie paragraaf 2.4) te weten:

- Business architectuur;
- Informatie architectuur;
- Technologische architectuur.

Binnen deze drie gebieden kunnen verschillende views gekozen worden. De definitie van enterprise architectuur zoals opgesteld is in paragraaf 2.1 gaat uit van 9 views: 4 op het gebied van de business, 2 op het gebied van de informatie en 3 op het gebied van de technologie. Net zoals complexiteit zich in het algemeen niet goed laat meten, laat complexiteit zich binnen enterprise architectuur ook niet meten op een algemene manier. Er zijn teveel variabelen om een algemene meetmethode te ontwikkelen. Naar mijn mening is de beste methode het vaststellen uit hoeveel distincties en connecties een view bestaat. Aan de hand daarvan kan een nieuwe (gewenste) situatie worden voorgesteld met minder distincties en connecties.

Om vast te stellen of er een kans bestaat dat complexiteit aanwezig is, kan er gebruik gemaakt worden van een klein heuristisch vragenlijstje zoals in Tabel 3 is opgesteld. Deze tabel maakt onderdeel uit van het probleemgebied dat besproken is in paragraaf 1.2.

Vraag	Antwoord
Vind u uw beheerkosten van de ICT te hoog in vergelijking met hetgeen zij biedt?	JA → kans op ICT complexiteit
Kunt u snel reageren op markt veranderingen?	NEE → kans op ICT complexiteit
Bent u in staat uw ICT strategie voldoende uit te voeren?	NEE → kans op ICT complexiteit
Heeft u voldoende overzicht over uw huidige ICT?	NEE → kans op ICT complexiteit

Tabel 3 complexiteitsvragen

Door het invullen van het raamwerk kan worden vastgelegd waar de complexiteit zich bevindt. Niet alle aspecten zijn belangrijk bij het constateren van complexiteit. “Appendix B: ORM-Modellen” en “Appendix C: Uitwerkingen systeemontwerp middels logisch geformuleerde dependency-structuren ” maken inzichtelijk welke aspecten van belang zijn bij complexiteit. Paragraaf 3.5.1 tot en met paragraaf 3.5.3 geven een overzicht van complexiteit op de drie niveaus van enterprise architectuur. Het systeemontwerp middels logisch geformuleerde dependency-structuren en de ORM-modellen zijn het uitgangspunt, het case bedrijf (appendix A: Case) zal ter illustratie dienen.

3.5.1 Complexiteit Business Architectuur

Op het niveau van de business wordt in grote lijnen uitgestippeld wat er van de ICT verlangd wordt. De diversiteit van het aantal gevoerde producten en diensten van de organisatie met de daarbijbehorende processen draagt bij aan de complexiteit. Per gebied zal worden aangegeven waar zich de complexiteit bevindt.

Product

Het geld binnen een organisatie wordt verdiend met het verkopen van producten of het leveren van diensten of beide. Producten van een organisatie kunnen ook bijdragen aan de complexiteit. Hoe meer verschillende soorten producten een organisatie levert (distincties) hoe lastiger het wordt om alles te blijven overzien. Een product kan verkocht worden aan verschillende soorten klanten. Het materiaal dat nodig is voor een product wordt geleverd door verschillende soorten leveranciers. Om een product te kunnen verkopen of inkopen zijn er verschillende processen nodig. Een product heeft dan ook een connectie met een of meerdere processen.
Om de complexiteit te kunnen vaststellen is er voldoende informatie nodig
E_Product (na) = E_product (voor) - I. De informatie betreffende de producten kan uit de documentatie over de producten gehaald worden en uit de organisatie zelf. Tabel 4 geeft de distincties en connecties weer voor het case bedrijf.

	Deelproduct	Klant	Leverancier	Proces
Distincties	Camera klein Camera groot Videorecorder A Videorecorder B Etc...	Particulier Zakelijk MKB Zakelijk BANK Etc..	Sony Fuji Philips Kodak Etc...	Inkoop VerkoopMKB VerkoopBANK VerkoopPART Etc...
Connecties	Camera klein wordt geleverd door Sony aan klant Particulier door proces verkoopPART Camera klein wordt geleverd door Philips door proces inkoop Camera groot wordt geleverd door Fuji aan klant Zakelijk MKB door proces verkoopMKB. Etc.....			

Tabel 4 product

Dienst

Naast het leveren van producten, leveren veel organisaties ook diensten of een combinatie van beide. Het aantal verschillende soorten diensten (distincties) dat een organisatie levert draagt bij aan de complexiteit. Diensten worden vaak afgestemd op een bepaald type klant. De diversiteit van de klanten (distinctie) draagt ook bij aan de complexiteit van de dienst. Daarnaast maakt een dienst nog gebruik van leveranciers die bepaalde onderdelen van een dienst leveren. Om de complexiteit van de dienst goed in kaart te kunnen brengen is er voldoende informatie nodig.
Om een dienst te kunnen leveren aan een klant zijn er bepaalde processen nodig die dit ondersteunen. Een dienst heeft daarom ook een connectie met proces. Tabel 5 geeft de distincties en connecties weer voor het case bedrijf.

	Deeldienst	Klant	Leverancier	Proces	Product
Distincties	Onderhoud_camera_klein Onderhoud_camera_groot 24uursbeveiliging Etc...	Particulier Zakelijk MKB Zakelijk BANK Etc..	Installateurs_Jansen Installateurs_Pietersen Secure Etc...	InkoopDienst VerkoopMKB VerkoopBANK VerkoopPART Etc...	Camera klein Camera groot Videorecorder A Videorecorder B Etc...
Connecties	Onderhoud_camera_klein wordt geleverd door Installateurs Jansen aan klant Particulier door proces verkoopPART 24uursveiliging wordt geleverd door Secure met product Videorecorder A door proces VerkoopMKB Onderhoud_camera_groot wordt geleverd door Installateurs Pietersen aan klant Zakelijk MKB door proces verkoopMKB. Etc...				

Tabel 5 dienst

Proces

Processen zorgen dat er producten en diensten geleverd kunnen worden. Binnen de processen staat omschreven welke handelingen uitgevoerd moeten worden om het uiteindelijk product of dienst te kunnen leveren. Een proces wordt daarom opgedeeld in procedures met daaronder een aantal werkinstructies die leiden tot concrete handelingen. Om processen goed te kunnen uitvoeren worden zij ondersteund door ICT. Het aantal verschillende soorten processen, procedures, werkinstructies en handelingen (distincties) draagt bij aan de complexiteit. Een proces heeft connecties met de diensten en producten en met de applicaties. Om de processen in kaart te kunnen brengen is er informatie nodig over de uitvoering van de processen.
Tabel 6 geeft de distincties en connecties weer van proces voor het case bedrijf.

	Proces	Procedure	Werkinstructie	Handeling	gegevens	Applicatie
Distincties	InkoopDienst VerkoopMKB VerkoopBANK VerkoopPART AdminPART Etc...	Onderhandelen Kopen Verkopen Boekhouden Etc..	Afrekenen Factureren Balansen Etc..	Contant Pinnen Invoeren Tellen Etc...	Bonnenboek Contracten Etc...	Ink_Dienst Ink_Product Verk_Dienst Verk_Product Baan Etc...
Connecties	proces verkoopPART bevat[1] procedure verkopen welke bestaat uit Werkinstructie afrekenen dat bestaat uit de handeling Contant, de Applicatie Verk_Dienst biedt ondersteuning en de van gegevens Bonnenboek wordt gebruik gemaakt. proces inkoopDienst bevat[1] procedure kopen welke bestaat uit Werkinstructie afrekenen dat bestaat uit de handeling Pinnen, de applicatie Ink_Product biedt ondersteuning. proces AdminPART bevat[1] procedure Boekhouden welke bestaat uit Werkinstructie factureren dat bestaat uit Handeling Invoeren, de applicatie Baan biedt ondersteuning. Etc.....					

Tabel 6 Proces

[1] Het woord bevat kan ook gelezen worden als: bestaat uit en / of heeft te maken met.

3.5.2 Complexiteit Informatie Architectuur

Op het niveau van de Informatie Architectuur wordt omschreven waar de applicaties aan moeten voldoen en wat ze moeten ondersteunen. Daarnaast wordt er bepaald welke gegevens nodig zijn. De diversiteit van het aantal gevoerde applicaties en gegevens draagt bij aan de complexiteit. Per gebied zal worden aangegeven waar de complexiteit zich bevindt en welke relaties er bestaan.

Applicatie

Applicaties bieden ondersteuning bij het uitvoeren van diverse processen. De applicaties zijn geschreven in diverse programmeertalen, zijn ontworpen volgens een bepaalde methode en werken met elkaar samen. Een applicatie ondersteunt altijd een bepaald doel dat door middel van een proces wordt gerealiseerd.
Daarnaast gebruikt een applicatie gegevens die uit een database komen. Het aantal applicaties (distincties) dat een organisatie heeft draagt bij aan de complexiteit. Om de complexiteit in kaart te brengen moet er informatie beschikbaar zijn. Het aantal verbindingen dat een applicatie heeft met andere objecten draagt ook bij aan de complexiteit. Tabel 7 Applicatie geeft de distincties en connecties weer van applicatie voor het case bedrijf.

	Applicatie	Prog. Taal	Platform	Middle-Ware	Ontwerp	Doel	Gegevens
Distincties	Ink_ Dienst Ink_ Product Verk_ Dienst Verk_ Product Baan Etc...	Pascal Cobol C++ Delphi Etc..	Windows XP (30) Windows 98 (32) Windows 95 (2) Unix 8.0 (2) AS/400 (1) Linux 7.0 (1) Etc....	IBM Websphere (1) Microsoft .Net (1) Etc...	Rup Yourdon Etc..	InkoopDIENST AdminPART Etc...	Klantenbestand Leverancier sbestand Verkoopbestand inkoopbestand Etc...
Connecties	Applicatie Ink_Dienst INK002 is ontworpen met RUP en gebouwd met Pascal. Applicatie Ink_Dienst INK002 gebruikt gegevens uit G001 Inkoopbestand DB2 en draait op Wxp_002 Windows XP Applicatie Ink_Dienst INK002 is ontworpen met RUP en gebouwd met Pascal. Applicatie Ink_Dienst INK002 gebruikt gegevens uit G001 Inkoopbestand DB2 en draait op W98_004 Windows 98 Applicatie Verk_Product VER001 is ontworpen met Yourdon en gebouwd met C++. Applicatie Verk_Product VER001 gebruikt gegevens uit G002 Verkoopbestand Oracle en maakt verbinding met M001 IBM Websphere en draait op W95_003 Windows 95. Etc.....						

Tabel 7 Applicatie

Gegevens
De gegevens van een organisatie vormen de kennis van de organisatie. Gegevens staan op allerlei plaatsen opgeslagen. Gedacht moet worden aan documenten en gegevensbanken. Om die gegevens nuttig te gebruiken en er daadwerkelijk wat mee te doen zijn er applicaties en processen. Applicaties gebruiken de gegevens. Gegevens vormen daarnaast ook de input voor processen.

De complexiteit van gegevens bestaat voornamelijk uit de diversiteit van gegevens (distincties) en de connecties die gegevens hebben met andere objecten. Om dit goed in kaart te brengen moet er informatie beschikbaar zijn over de gegevens. Tabel 8 Gegevens geeft de distincties weer van gegevens voor het case bedrijf.

	Document	Gegevensbestand	Platform
Distincties	Kwaliteitshandboek Bonnenboek Contracten Orders Etc....	Klantenbestand Leveranciersbestand Verkoopbestand inkoopbestand Etc....	Windows XP (30) Windows 98 (32) Windows 95 (2) Unix 8.0 (2) AS/400 (1) Linux 7.0 (1) Etc....
Connecties	Zie applicatie en proces voor de connecties		

Tabel 8 Gegevens

3.5.3 Complexiteit Technologische Architectuur

De technologie architectuur bestaat uit de Middleware, de platformen en de netwerken van een organisatie. Een groot gedeelte van deze complexiteit is te danken aan het verleden, zie Paragraaf 3.3. In deze paragraaf zullen de drie technologische aspecten behandeld worden en zal er worden aangegeven waar de complexiteit zich bevindt.

Middleware

De functie van middleware is het koppelen van verschillende soorten systemen en applicaties om deze met elkaar te kunnen laten communiceren. Middleware is het best te vergelijken met dubbelzijdig plakband en fungeert als een soort van brug tussen systemen onderling en applicaties onderling.
Om de complexiteit van middleware binnen enterprise architectuur in kaart te brengen geldt het aantal soorten middleware dat gebruikt wordt (type) en het aantal van dat type. Om dat goed te kunnen doen is er voldoende informatie nodig over middleware.
Daarnaast moet het aantal verbindingen in kaart worden gebracht. Middleware heeft een verbinding met de verschillende applicaties en de platformen (onderdeel systeemsoftware). Tabel 9 Middleware zal laten zie hoe de complexiteit van middleware achterhaald kan worden.

	Middleware
Distincties	IBM Websphere (1) Microsfoft .Net (1) Etc...
Connecties	M001 IBM Websphere Wxp_002 Windows XP M001 IBM Websphere W95_003 M002 Microsfoft .Net AS_001 AS/4000 Etc.....

Tabel 9 Middleware

Platform

Een platform behandelt de gebruikte hardware en systeemsoftware van een organisatie. De hardware kan bestaan uit pc's, mini's en mainframes. Op die verschillende hardware draait systeemsoftware. Systeemsoftware is van een bepaald type. De hardware heeft een verbinding met het netwerk, en de systeemsoftware heeft verbinding met één of meerdere applicaties en kan daarnaast nog verbinding hebben met middleware. Het is belangrijk om voldoende informatie te verkrijgen over het platform om de complexiteit te kunnen vaststellen. Het eerste wat vastgesteld kan worden is het aantal verschillende type hardware (distincties) en het aantal verschillende soorten systeemsoftware.
Vervolgens kan bekeken worden hoeveel connecties er tussen de objecten zijn.
Het platform is leidend voor het netwerk, er zal in Tabel 10 Platform ook worden opgenomen wat de verbindingen zouden kunnen zijn met het netwerk.

	Hardware	Systeemsoftware
Distincties	PC Dell 800mhz (24) PC Dynabyte 2,4ghz (40) Mini M12B IBM (3) Mainframe MF3400 (1) Etc....	Windows XP (30) Windows 98 (32) Windows 95 (2) Unix 8.0 (2) AS/400 (1) Linux 7.0 (1) Etc....
Connecties	PC P003 Dell 800mhz Wxp_002 Windows XP PC P003 Dell 800mhz Hub H002 (sweex 8p) UTP 15 PC P006 Dynabyte 2,4ghz W95_003 Windows 95 PC P006 Dynabyte 2,4ghz Hub H004 (Cisco 16p) UTP 10 Mf Mf001 MF3400 AS_001 AS/4000 Mf Mf001 MF3400 Router R003 3com 4p (6) Glasvezel 2 Etc.....	

Tabel 10 Platform

Netwerk
Binnen de view netwerk wordt in kaart gebracht met welke infrastructuur de organisatie werkt. Om de complexiteit daarvan in kaart te brengen moet er eerst bepaald worden waar een netwerk uit opgebouwd is. Een netwerk bestaat uit routers, switches, hubs en bekabeling. Een netwerk bestaat uiteraard uit nog meer onderdelen, die zullen in dit onderzoek niet behandeld worden, maar kunnen later altijd nog worden toegevoegd.
Om de complexiteit te kunnen vaststellen is het belangrijk om voldoende informatie over het netwerk te hebben. Om dit te bereiken kan de formule van entropie worden gebruikt. E_netwerk(na) = E_netwerk(voor) - I. Aan de hand van de verkregen informatie moet naar voren komen hoeveel verschillende type (distincties) routers, switches, hubs en bekabeling er in omloop zijn. Daarnaast is het belangrijk om te weten hoeveel routers, switches, hubs en bekabeling er per type zijn. Het aantal connecties kan bepaald worden door te onderzoeken hoeveel verbindingen een object heeft met andere objecten. Tabel 11 Netwerk geeft een overzicht hoe dat eruit zou kunnen zien.

	Hub	Router	Switch	Bekabeling
Distincties	Sweex 8p (1) Cisco 16p (1) 3com 4p (2) Etc...	Cisco 8p (2) Etc...	Sweex 16p (4) Etc....	UTP (500) Glazvezel (5) BNC (10) Etc...
Connecties	Hub H002 (sweex 8p) is verbonden met Hub H004 (Cisco 16p) door middel van kabel UTP (23) Hub H003 (cisco 16p) is verbonden met Switch S003 (Sweex 16p) door middel van kabel UTP (306) Etc.....			

Tabel 11 Netwerk

3.6 Organisatie in relatie met complexiteit

Complexiteit speelt zich af binnen een organisatie, omdat er verschillende soorten organisaties zijn is er ook sprake van verschillende omvangen van complexiteit. Voordat wordt overgegaan tot het detecteren en reduceren van complexiteit is het goed stil te staan bij het type organisatie.
Intuïtief is er bij de meeste mensen bekend wat onder een organisatie verstaan wordt. Net als bij veel zaken die vanzelfsprekend lijken is het begrip organisatie moeilijk te definiëren. Dit wordt voor een deel veroorzaakt door de verschillende manieren waarop mensen naar organisaties kijken. Juristen zien een organisatie als rechtspersoon met bepaalde rechten en plichten. Bedrijfseconomen zien een samenwerkingsverband waarin mensen doelbewust, rationeel en berekenend handelen. Een Informatiekundige kijkt naar een organisatie in termen van gegevensstromen en informatie-uitwisseling tussen bedrijfsprocessen en mensen. Algemeen kan een organisatie als volgt omschreven worden:

> Een organisatie is een herkenbare eenheid, waarin mensen op gecoördineerde wijze met behulp van technische en financiële middelen activiteiten uitvoeren, teneinde gemeenschappelijke doelen te realiseren

[PASCOE]

Uit deze definitie komen drie aspecten voor:

- De organisatie als maatschappelijke eenheid;
- De organisatie in de zin van structuur, waarin de taken zijn verdeeld;
- De organisatie als proces, waarin de activiteiten op gecoördineerde wijze worden uitgevoerd.

Iedere organisatie is anders ingericht om aan deze drie aspecten te voldoen. Om een beeld te krijgen van de type organisaties zal ingegaan worden op de verschillende types.

Iedere organisatie steekt anders in elkaar. Sommige organisaties zijn sterk hiërarchies opgebouwd, andere organisaties zijn informeel ingericht. Om onderscheid te kunnen maken tussen de verschillende organisaties is het nodig ze te typeren. Een bepaald type organisatie heeft te maken met een bepaalde complexiteit, of is gevoeliger voor een bepaalde complexiteit. Er zijn vier belangrijke contingentiefactoren te onderscheiden die invloed hebben op de organisatie, te weten:

- De omgeving;
  - Stabiliteit;
  - Vijandigheid;
  - Marktdiversiteit;
- De technologie;
  - Kennisniveau;
  - Productietechnieken;
- De leeftijd en omvang;
  - Leeftijd;
  - Omvang;
- Machtsaspecten.

3.6.1 Type Organisatie

Aan de hand van de hierboven omschreven contingentiefactoren kan een indeling gemaakt worden qua type organisatie. Er is gekozen voor de typologie van Mintzberg, aangezien Mintzberg een van de grondleggers op het gebied van de organisatiekunde is. Mintzberg is van mening dat het niet alleen om enkelvoudige relaties tussen de contingentievariabelen gaat, maar om een configuratie van contingentievariabelen. Er zijn volgens Mintzberg zeven configuraties, te weten:

- De ondernemersorganisatie;
- De machineorganisatie;
- De gediversifieerde organisatie;
- De professionele organisatie;
- De innovatieve organisatie;
- De zendingsorganisatie;
- De politieke organisatie;

De ondernemersorganisatie

Een ondernemersorganisatie wordt gekenmerkt door een kleine omvang, meestal 1 tot 2 mensen die de macht hebben binnen de organisatie. Een ondernemersorganisatie heeft ICT vaak als ondersteunend middel, bestaande uit een aantal standaard pakketen, qua omvang is de ICT vaak gering. De ICT wordt vaak zelf gedaan of door gespecialiseerde bedrijven voor het MKB. De expertise op het gebied van ICT is vaak gering.

Voorbeelden van ondernemersorganisaties zijn onder andere :

- Veel MKB bedrijven;
- Kleine Dienstverleners;

Machineorganisatie

Een machineorganisatie is een grote, relatief volwassen organisatie. Kenmerkend voor dit soort organisaties zijn de routinematige, relatief eenvoudige taken in de operationele kern. Binnen een machineorganisatie is er vaak een eigen ICT afdeling die het grootste gedeelte van de ICT regelt. De omvang van de ICT is groot en er dienen daarom duidelijke beheerplannen te liggen om de ICT adequaat te kunnen controleren. Binnen dit soort organisaties zijn vaak grote standaardpakketten geïnstalleerd (ERP,CRM etc.). Omdat machineorganisaties vaak al wat oudere organisaties zijn bestaat er ook een diversiteit aan ICT zie paragraaf 3.3.
Voorbeelden van dit soort type organisaties zijn onder ander:

- Fabrieken.

Professionele organisatie

Professionele organisaties beschikken over het algemeen over hoogopgeleide medewerkers in de operationele kern. Het werk dat zij uitvoeren is vrij complex, er geldt daarnaast een zekere mate van standaardisatie. De ICT heeft een puur ondersteunende rol, met name werkplekken met de benodigde software. Naarmate de organisatie qua omvang toeneemt en er verschillende locaties zijn, neemt de ICT complexiteit toe. De medewerkers beslissen vaak zelf wat zij nodig achten voor hun werkzaamheden, hierdoor is het moeilijk om een goed overzicht te krijgen van de aanwezig ICT.
Voorbeelden van dit soort organisaties zijn:

- Dienstverlenende sector;
- Scholen;
- Software-huizen.

Innovatieve Organisatie

Innovatieve organisaties zijn vaak jonge organisaties met een lage formalisatiegraad en flexibel ingericht. Kenmerkend voor dit soort organisaties is het multidisciplinair werken in projectteams. De technieken die gebruikt worden zijn vaak complex. Om dit goed te ondersteunen zijn er speciale ICT voorzieningen nodig. ICT systemen worden geleverd door gespecialiseerde bedrijven. De ICT zal in dit soort organisaties snel veranderen, om dit mogelijk te maken is een flexibele ICT indeling noodzakelijk.
Voorbeelden van innovatieve organisaties zijn:

- Biotechnische onderzoeksbureaus;
- Research Bedrijven;
- Ruimtevaart ondernemingen.

Gediversifieerde organisatie

Vaak wat oudere organisaties met diverse producten of diensten, met verschillende typen klanten of opereren in verschillende regio's zijn vaak gediversifieerde organisaties. Hiërarchies beststaat een gediversifieerde organisatie uit een aantal divisies die onder het gezamenlijk bestuur van een hoofdkantoor vallen.
Er wordt een ICT beleid geformuleerd op divisie niveau, zoals het ook bij een machineorganisatie plaatsvindt. Daarnaast heeft het hoofdkantoor vaak regels opgesteld waar een divisie zich aan dient te houden. Het resultaat hiervan is dat er een diversiteit aan ICT ontstaat, er binnen aparte divisies min of meer dezelfde ICT applicaties zijn, maar net weer anders.
Voorbeelden van Gediversifieerde organisaties zijn:

- Grote elektronica bedrijven;
- Levensmiddelen organisaties;
- Productiebedrijven.

Zendingsorganisaties

Een zendingsorganisatie is ontstaan uit een ideaal. De medewerkers delen allemaal hetzelfde ideaal. ICT kan hierbij een rol spelen. In dit soort organisaties staat het vaak op een laag pitje en is er bovendien bijna geen budget voor.
Voorbeelden van zendingsorganisaties zijn:

- Religieuze instellingen;
- Liefdadigheidsstichtingen.

Politieke organisatie

Een politieke organisatie is een organisatie waar veel conflicten heersen. Er zit geen structuur meer in de organisatie, geen autoriteit, ideologie of expertise. Op het gebied van ICT is er ook geen duidelijk beleid meer, degene die het voor het zeggen heeft bepaalt wat er gebeurt. Er zijn geen specifieke voorbeelden van organisaties die een politieke organisatie zijn. Dit is meestal een gevolg van conflicten binnen een organisatie.

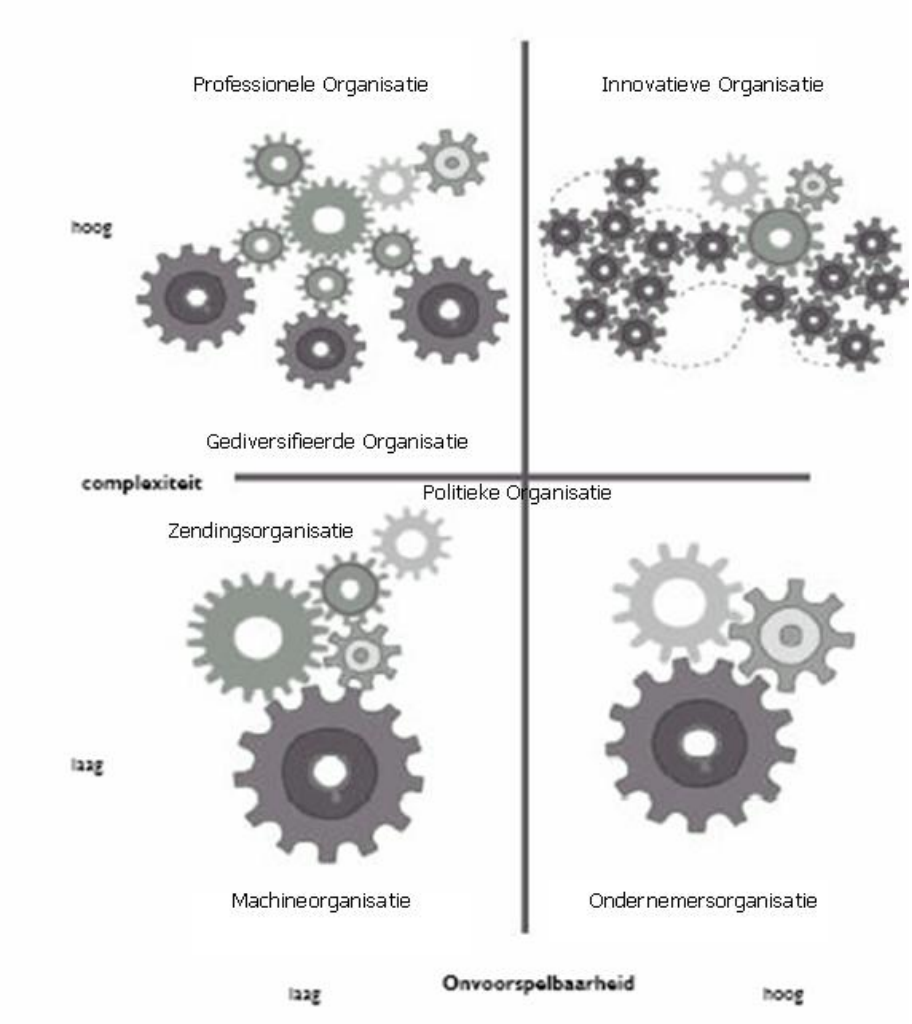
[MINTZ] [PASCOE]

3.6.2 Overzicht Complexiteit en type organisatie

De behandelde type organisaties kunnen worden gerangschikt naar de complexiteit van een organisatie en naar de voorspelbaarheid van de markt zie Figuur 10. Dat wil zeggen hoe flexibel kan een organisatie reageren op markt veranderingen, en hoeveel last heeft het van complexiteit? (zie Figuur 8.)

Type Organisatie	Voorspelbaarheid markt	Complexiteit
Machineorganisatie	Goed (stabiel)	Laag
Zendingsorganisatie	Goed	Laag
Gediversifieerde Organisatie	Goed (stabiel)	Hoog (divers)
Professionele Organisatie	Redelijk stabiel	Hoog
Innovatieve Organisatie	Laag	Hoog
Ondernemersorganisatie	Laag (dynamische)	Laag
Politieke Organisatie	N.V.T.	N.V.T.

Tabel 12 Organisatie overzicht



Figuur 10 Complexiteit Organisatie

3.7 Samenvatting

Complexiteit bestaat uit twee kerndelen, namelijk: het aantal distincties (verschijningen van een object) en het aantal connecties (verbindingen tussen de objecten). Als er sprake is van veel distincties en veel connecties wordt er gesproken over complexiteit. Complexiteit is tot op heden niet op een eenduidige manier meetbaar, de voornaamste reden hiervoor is dat ieder verschijnsel anders is en niet zomaar te vergelijken valt met een ander verschijnsel. De lengte van de kortste beschrijving van een systeem is tot nu toe de meest gebruikte vorm om complexiteit te meten. Het idee hierachter is: hoe complexer het systeem is hoe lastiger het is om in zijn geheel te beschrijven. Hierbij speelt het aspect informatie een belangrijke rol. Als er niet bekend is wat het systeem doet: er is onvoldoende informatie over dat systeem; kan er ook niks zinnigs gezegd worden betreffende de complexiteitsgraad.

De statische toestand van complexiteit is echter niet erg interessant. Belangrijker is de verandering in complexiteit: neemt het toe of af? Complexiteit neemt toe als het aantal distincties of connecties toeneemt en neemt af als het aantal distincties of connecties afneemt.

Complexiteit is vaak ontstaan uit het verleden, ICT is intussen sterk gegroeid en groeit nog steeds zonder dat er daadwerkelijk zicht op is waar het voor dient. De afstemming tussen de business en de ICT speelde in het verleden nog geen belangrijke rol. De verwachting is dat in de toekomst de ICT nog harder en verder zal groeien.

Complexiteit binnen organisaties heeft ook een aantal negatieve gevolgen: 79% van de bedrijven geeft aan dat complexiteit de oorzaak is voor het niet bereiken van ICT strategische doelstellingen. Meer dan de helft van die bedrijven is het daadwerkelijk niet gelukt om de ICT doelstellingen te realiseren. Een ander belangrijk gevolg van de ICT complexiteit is de time-to-market. Organisaties zijn niet meer in staat snel te reageren op marktveranderingen omdat de ICT te complex is geworden. Daarnaast nemen door de complexiteit de onderhouds- en beheerkosten sterk toe. Kleine functionele wijzigingen in programma's hebben enorme implementatiekosten tot gevolg vanwege de complexiteit. Als laatste komt de continuïteit van de organisatie zelf in gevaar.

Aan de hand van complexiteitsvragen kan worden bekeken of er een kans bestaat dat een organisatie te maken heeft met ICT complexiteit. Het enterprise architectuur raamwerk maakt de complexiteit inzichtelijk. Per view moet naar voren komen uit hoeveel voor de view relevante distincties en connecties deze bestaat. De ORM-modellen en het systeemontwerp middels logisch geformuleerde dependency-structuren leggen formeel vast welke aspecten ter zake doen.

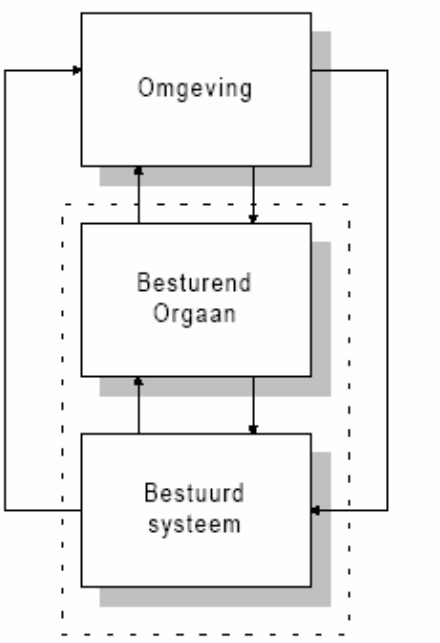
Verschillende type organisaties hebben te maken met verschillende omvangen van complexiteit. Per type organisatie kan globaal worden vastgelegd welke complexiteitsaspecten een rol spelen. Binnen dit onderzoek is gekozen voor de typering van Mintzberg en is per type organisatie vastgelegd wat de mate van complexiteit is.

4 Beïnvloeden van complexiteit

Dit hoofdstuk zal pogen inzicht te verschaffen in het sturen en het beïnvloeden van complexiteit. Allereerst zal worden stilgestaan bij het begrip sturen en wat men er mee wil bereiken. Vervolgens zal er behandeld worden hoe de complexiteit daadwerkelijk gereduceerd kan worden.

4.1 Sturen

Om de complexiteit te kunnen reduceren moet de complexiteit worden beïnvloed. De complexiteit moet een bepaalde richting opgaan (sturen), teneinde de complexiteit te verminderen. Om dat te realiseren moet er een orgaan zijn dat de rol van bestuurder vervult en invloed kan uitoefen op complexiteit.
Het besturingsparadigma maakt dit inzichtelijk. Het besturingsparadigma van de Leeuw (zie Figuur 11) vertegenwoordigt een kijk op besturingssituaties.

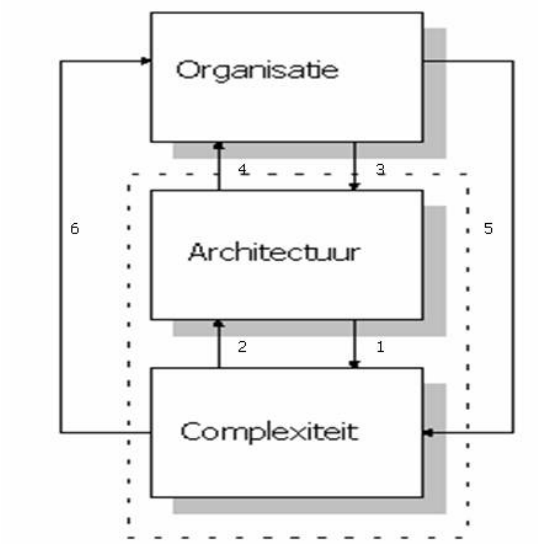


Figuur 11 Besturingsparadigma [LEEUW]

Het besturingsparadigma is een interactie tussen drie deelsystemen te weten:

- Het Bestuurd systeem: het deelsysteem waarmee de gewenste bijdrage tot stand moet worden gebracht;
- De omgeving: het deelsysteem waaraan de gewenste bijdrage wordt geleverd;
- Het besturend orgaan: het deelsysteem dat bepaalde doelen wil bereiken door het bestuurde systeem gericht te beïnvloeden.

Wordt het besturingsparadigma binnen dit onderzoek toegepast, om complexiteit te willen verminderen, dan kan er het volgende diagram gemaakt worden, zie Figuur 12.



Figuur 12 Complexiteit Paradigma

- Complexiteit: vervult de rol van bestuurd systeem. Dit systeem moet beïnvloed worden om de gewenste bijdrage te leveren;
- Organisatie: vervult de rol van omgeving, aan de organisatie wordt de gewenste bijdrage geleverd, namelijk minder complexiteit;
- Architectuur: vervult de rol van besturend orgaan, om de complexiteit gericht te beïnvloeden. Architectuur zelf stuurt niet, maar wordt gebruikt door de bestuurder om te sturen. De ICT architect is in de meeste gevallen de bestuurder die het middel architectuur gebruikt om invloed uit te oefenen.

De ICT architect gebruikmakend van architectuur probeert zijn / haar doel te bereiken door de complexiteit en de organisatie d.m.v. stuurmaatregelen te beïnvloeden, zie nummers 1 en 4 in Figuur 12.
Om gericht te kunnen sturen moet de ICT architect gebruikmakend van architectuur weten hoe de complexiteit er uitziet en de situatie van de organisatie is, zie nummers 2 en 3 in Figuur 12.
De pijlen 5 en 6 geven de bijdrage weer: de uiteindelijke toegevoegde waarde van complexiteit aan de organisatie, welk resultaat levert de complexiteit aan de organisatie en wat heeft de complexiteit daarvoor nodig van de organisatie.
Twee belangrijke factoren die van invloed zijn op het uiteindelijke resultaat zijn het besturend vermogen van de ICT architect gebruikmakend van architectuur, en de bestuurbaarheid van complexiteit.

4.2 Bestuurbaarheid

Onder bestuurbaarheid kan het volgende worden verstaan:

> De mate waarin er stuurmaatregelen bestaan, de mate waarin complexiteit te beïnvloeden is.

[PANFOX]

Onder besturend vermogen kan het volgende worden verstaan:

> Het vermogen van architectuur om tijdig de juiste stuurmaatregelen te kiezen en toe te passen, de mate waarin architectuur vaardig is in het hanteren van de besturingssituatie.

[PANFOX]

Voor een effectieve besturing moet architectuur over de volgende zaken beschikken:

- Kennis van de doelen, met inbegrip van een mechanisme ter beoordeling van het effect van de besturing;
- Een model van de complexiteit en de organisatie om het effect van de stuurmaatregel te kunnen voorspellen;
- Informatie over de actuele toestand van complexiteit en organisatie;
- Voldoende stuurmaatregelen;

[PANFOX]

4.3 Complexiteitsreductie

Architectuur moet beschikken over een aantal stuurmaatregelen om complexiteit en de organisatie te kunnen beïnvloeden. De pijlen van Figuur 12 zullen behandeld worden om aan te geven wat er nodig is om complexiteit te reduceren.

Pijl 2
Met behulp van het raamwerk kan er per view worden afgelezen waar de complexiteit zich bevindt. De ORM-modellen en het systeemontwerp middels logisch geformuleerde dependency-structuren omschrijven wat hier voor nodig. Paragraaf 3.5 behandelt complexiteit in relatie met een raamwerk uitgebreid.

Pijl 3
Architectuur wil uiteindelijk invloed uitoefenen op een organisatie. Om dat te kunnen doen moet er bekeken worden van welk type de organisatie is. Paragraaf 3.6 beschrijft welke soorten organisaties er zijn, en met welke vorm van complexiteit zij te maken hebben.

Pijl 1
Met behulp van de verkregen informatie over complexiteit en de organisatie kan de complexiteit gericht worden beïnvloed. Het is zaak om per view de distincties en connecties te verminderen. Bijvoorbeeld het aantal operating systemen zien terug te brengen, of de verschillende soorten databases te verminderen. Er dienen projecten te worden geformuleerd waarin de aanpak staat omschreven die de complexiteit van een bepaalde view vermindert. Projecten die de complexiteit verminderen zijn bijvoorbeeld:

- 1 type netwerk, zelfde switches, routers, hub's en bekabeling. Als het aantal type van de switches, routers, hub's en bekabeling wordt gereduceerd neemt de complexiteit af. (Netwerk view)
- Het reduceren van platformen, 1 standaard werkplek definiëren en deze bedrijfsbreed doorvoeren. Hierdoor neemt het overzicht toe en hoeven aanpassingen niet meer platform specifiek te worden gemaakt (Platform view)
- Kiezen voor 1 middleware standaard met een grote mate van flexibiliteit, welke goed samenwerkt met de applicaties, gegevens, platformen en netwerk. (middleware view)
- Het aantal gegevensbestanden omzetten naar 1 standaard. Deze standaard moet ondersteund worden door de applicaties, middleware en de platformen. (gegevens view)
- Bundelen van de applicaties, applicaties met min of meer dezelfde functionaliteit koppelen. Het aantal afhankelijkheden van applicaties zien te verminderen. (applicatie view).
- Het aantal processen zien te verminderen en de afhankelijkheden van processen onderling te reduceren. Processen zo inrichten dat ze op een gestandaardiseerde manier ondersteund kunnen worden door ICT (proces view).
- Het aantal verschillende soorten producten en diensten laten afnemen. Een flexibel product of dienst introduceren, dat gemakkelijk aangepast kan worden en ondersteund kan worden door een beperkt aantal processen en applicaties (product view, dienst view).

Pijl 4
Door gebruik te maken van de opgestelde projecten die de complexiteit reduceren kan de gewenste bijdrage aan de organisatie geleverd worden. Er ontstaat na uitvoering van het project een organisatie met minder complexiteit.

Pijl 5
Het soort organisatie draagt bij aan een bepaalde mate van complexiteit zie paragraaf 3.6.

Pijl 6
Complexiteit zorgt ervoor dat de organisatie minder goed in staat is te opereren, zie paragraaf 3.4.

4.4 Samenvatting

Om complexiteit uiteindelijk te reduceren moet er invloed worden uitgeoefend op de complexiteit. Binnen dit onderzoek is hiervoor gebruik gemaakt van het besturingsparadigma van de Leeuw.

Complexiteit (bestuurd systeem) kan beïnvloed worden door een besturend orgaan, in het onderzoek is de architect die gebruik maakt van enterprise architectuur het besturend orgaan. Om de complexiteit te verminderen is het zaak per view het aantal connecties en distincties te verminderen. De omgeving waar een bijdrage aan geleverd wordt is de organisatie. Na het uitvoeren van projecten die complexiteit reduceren ontstaat er een organisatie met minder complexiteit.

5 Conclusie

Binnen dit hoofdstuk zullen de conclusies per deelvraag behandeld worden om uiteindelijk antwoord te geven op de hoofdvraag : *Hoe kan complexiteit inzichtelijk worden gemaakt en worden gereduceerd, binnen organisaties, door gebruik te maken van enterprise architectuur?*

5.1 Wat is enterprise architectuur?

Aan de hand van de geraadpleegde literatuur binnen dit onderzoek kan geconcludeerd worden dat binnen dit onderzoek het volgende onder enterprise architectuur verstaan wordt: Architectuur is het consistente geheel aan principes en modellen dat richting geeft aan ontwerp, realisatie en sloop van processen, organisatorische inrichting, informatievoorziening en technische infrastructuur op zowel het niveau van de business, de informatievoorziening als het technische niveau. Het werken onder architectuur zorgt ervoor dat losse onderdelen in hun samenhang worden ontworpen.

Enterprise architectuur levert een raamwerk op, welke bestaat uit een aantal viewpoints op een organisatie. Het enterprise architectuurraamwerk bestaat uit drie belangrijke componenten: een business-, een informatie- en een techniekraamwerk. Deze drie componenten belichten de drie belangrijkste aspecten van een organisatie.

Daarnaast is enterprise architectuur geen doel op zich maar een middel om bepaalde doelen te behalen. Binnen dit onderzoek heeft enterprise architectuur gediend als het middel. Ook is enterprise architectuur door middel van ORM en logisch geformuleerde dependency-structuren getracht in kaart te brengen.

5.2 Wat is het nut van enterprise architectuur?

Enterprise architectuur moet bepaalde doelen dienen wil het nuttig zijn. De belangrijkste doelen waar enterprise architectuur aan kan bijdragen volgens de geraadpleegde literatuur zijn:

- Complexiteitsreductie;
- Kostenbesparing;
- Bevordering van de afstemming tussen de ICT en de business;
- Communicatiemiddel.
-

Dit onderzoek heeft zich bezig gehouden met complexiteitsreductie.

5.3 Wat is een organisatie?

Een organisatie kan volgens de literatuur, zoals geraadpleegd is binnen dit onderzoek als volgt worden omschreven:
Een organisatie is een herkenbare eenheid, waarin mensen op gecoördineerde wijze met behulp van technische en financiële middelen activiteiten uitvoeren, teneinde gemeenschappelijke doelen te realiseren.

Hieruit komen drie aspecten voor:

- De organisatie als maatschappelijke eenheid;
- De organisatie in de zin van structuur, waarin de taken zijn verdeeld;
- De organisatie als proces, waarin de activiteiten op gecoördineerde wijze worden uitgevoerd.

5.4 Welke type organisaties vanuit ICT oogpunt zijn er?

Als er gekeken wordt vanuit ICT oogpunt naar een organisatie kan er geconcludeerd worden dat er zeven verschillende type organisaties zijn. Deze zeven verschillende type organisaties zijn vastgesteld aan de hand van de contingentiefactoren van Mintzberg, te weten:

- De ondernemersorganisatie;
- De machineorganisatie;
- De gediversifieerde organisatie;
- De professionele organisatie;
- De innovatieve organisatie;
- De zendingsorganisatie;
- De politieke organisatie;

5.5 Wat is complexiteit?

Binnen dit onderzoek is complexiteit onderzocht in zijn algemeenheid vanuit de literatuur en is er een vertaling gemaakt naar de ICT. Complexiteit wordt binnen dit onderzoek gedefinieerd als: de distincties (onderdelen) en connecties (verbindingen) die onderling een relatie hebben. Iets wordt als complex beschouwd als er geen inzicht of overzicht meer is in de distincties en connecties.

Binnen de ICT wordt complexiteit dan ook gezien als het aantal verschillende distincties en connecties per view afgeleid uit de enterprise architectuur.

5.6 Hoe is complexiteit af te lezen?

Uit het onderzoek en de geraadpleegde literatuur is ondervonden dat complexiteit kan worden afgelezen door gebruik te maken van een enterprise architectuur raamwerk. De opgestelde ORM-modellen en het systeemontwerp middels logisch geformuleerde dependency-structuren maken inzichtelijk welke onderdelen betreffende complexiteit nodig zijn om het enterprise architectuur raamwerk in te vullen. Uit het enterprise architectuur raamwerk kan vervolgens gekeken worden naar het aantal distincties en connecties per view.

5.7 Is complexiteit meetbaar?

Uit het onderzoek en het raadplegen van verschillende bronnen is gebleken dat complexiteit zich niet op een eenduidige manier laat meten. Er zijn in het verleden verschillende pogingen gedaan om complexiteit kwantificeerbaar te maken. Echter is geen van de pogingen algemeen bruikbaar. De definitie die het beste toe te passen is voor een systeem dat niet wiskundig of formeel is vastgelegd is: de lengte van de kortste omschrijving van het systeem. Het idee hierachter is: hoe complexer het systeem is hoe lastiger het is om in zijn geheel te omschrijven.

5.8 Hoe is ICT complexiteit ontstaan?

Naar aanleiding van onderzoek naar de geschiedenis van de ICT kan er geconcludeerd worden dat de ICT complexiteit is ontstaan uit het verleden. De ICT is in de jaren tachtig sterk gaan groeien, zonder dat er voldoende overleg is gevoerd tussen de organisatieafdelingen. De automatisering werd in die tijd bepaald door de ICT afdeling zelf, zonder dat er vaak sprake was van een bedrijfskundige onderbouwing. Het verbinden van de systemen met elkaar heeft bijgedragen aan de toename van complexiteit. Organisaties hebben nu systemen staan uit verschillende tijdperken die met elkaar zijn verbonden waar geen zicht meer op is. ICT complexiteit kan dus gezien worden als een erfgoed uit het verleden.

5.9 Wat zijn de gevolgen van ICT complexiteit?

Als er gekeken wordt naar de gevolgen van de ICT complexiteit kan er geconcludeerd worden dat het uitvoeren van de ICT strategie sterk beïnvloed wordt door de complexiteit. Onderzoek heeft aangetoond dat maar liefst 79% van de grote en middelgrote organisaties aangeeft dat de groeiende complexiteit de voornaamste oorzaak is om de ICT strategie niet voldoende te kunnen uitvoeren. Een ander belangrijk gevolg van de ICT complexiteit is de toename van de time-to-market. Organisaties zijn niet meer in staat snel te reageren op marktveranderingen omdat de ICT te complex is geworden. Daarnaast nemen door de ICT complexiteit de onderhouds- en beheerkosten sterk toe. Kleine functionele wijzigingen in programma's hebben enorme implementatiekosten tot gevolg vanwege de complexiteit. Als laatste komt de continuïteit van de organisatie zelf in gevaar.

5.10 Hoe kan complexiteit gereduceerd worden?

Om complexiteit uiteindelijk te reduceren moet er invloed worden uitgeoefend op de complexiteit. Binnen dit onderzoek is hiervoor gebruik gemaakt van het besturingsparadigma van de Leeuw.

Complexiteit (bestuurd systeem) kan beïnvloed worden door een besturend orgaan, in het onderzoek is de architect die gebruik maakt van enterprise architectuur het besturend orgaan. Om de complexiteit te verminderen is het zaak per view het aantal connecties en distincties te verminderen. De omgeving waar een bijdrage aan geleverd wordt is de organisatie. Na het uitvoeren van projecten die complexiteit reduceren ontstaat er een organisatie met minder complexiteit.

5.11 Hoe kan complexiteit in kaart worden gebracht met enterprise architectuur?

Aan de hand van de geraadpleegde literatuur en onderzoek kan complexiteit in kaart worden gebracht door gebruik te maken van een enterprise architectuur raamwerk. Het EAR is een middel en kan per view de complexiteit in kaart brengen. Om te weten waar de complexiteit uit bestaat wordt er binnen dit onderzoek gebruik gemaakt van een methodologie. De methodologie omschrijft hoe gebruikmakend van ORM en systeemontwerp middels logisch geformuleerde dependency-structuren complexiteit in kaart kan worden gebracht. De modellen maken inzichtelijk waar de complexiteit per view uit bestaat.

5.12 Welke relatie bestaat er tussen complexiteit en organisatie?

Per type organisatie zoals opgesteld aan de hand van de contingentiefactoren van Mintzberg bestaat er een andere omvang van complexiteit. De belangrijkste ondervindingen aan de hand van het onderzoek zijn hieronder opgenomen:

- Een machineorganisatie heeft een lage mate van complexiteit;
- Een Zendingsorganisatie heeft een lage mate van complexiteit;
- Een gediversifieerde organisatie heeft een hoge mate van complexiteit;
- Een professionele organisatie heeft een hoge mate van complexiteit;
- Een innovatieve organisatie heeft een hoge mate van complexiteit;
- Een ondernemersorganisatie heeft een lage mate van complexiteit.

5.13 Antwoord op de hoofdvraag

Aan de hand van de voorgaande deelvragen zal getracht worden een antwoord te geven op de hoofdvraag binnen dit onderzoek.

Hoe kan complexiteit inzichtelijk worden gemaakt en worden gereduceerd, binnen organisaties, door gebruik te maken van enterprise architectuur?

Uit het onderzoek is gebleken dat complexiteit inzichtelijk gemaakt kan worden door gebruik te maken van een enterprise architectuur raamwerk. De ORM-modellen en het systeemontwerp middels logisch geformuleerde dependency-structuren maken inzichtelijk hoe complexiteit is opgebouwd per view op de organisatie. De organisatie wordt door negen verschillende views bekeken. Iedere view behandelt een ander aspect van de organisatie. Aan de hand van de negen ingevulde views van het EAR is inzichtelijk gemaakt waar de complexiteit zich binnen een organisatie bevindt. Het type organisatie speelt ook een rol, elk type organisatie heeft te maken met een andere omvang van complexiteit.

Het reduceren van complexiteit bestaat uit het reduceren van het aantal distincties en connecties per view. Door gebruik te maken van enterprise architectuur kan per view een project worden geformuleerd die omschrijft hoe de connecties en distincties per view naar beneden gebracht kunnen worden. Na het uitvoeren van één of meer projecten ontstaat er naar alle waarschijnlijkheid een organisatie met minder complexiteit.

Appendix A: CASE

Complexi B.V. is een organisatie met ongeveer 850 hoger opgeleide medewerkers. Het bedrijf levert beveiligingsinstallaties aan diverse soorten klanten. Complexi B.V. kan getypeerd worden als een innovatieve organisatie. De markt waar zij in opereren verandert de laatste tijd sterk. De grootste oorzaak hiervan zijn de terroristische dreigen. Complexi B.V. Heeft daarom ook moeite om op de marktveranderingen in te spelen. Bij het lanceren van een nieuwe dienst hebben ze vaak moeite dit snel te doen. Ook zien zij al jarenlang de beheerkosten stijgen. Het managent gaf onlangs aan dat de ICT strategie niet meer de gewenste resultaten opleverde.

Complexi B.V. koopt op het moment de producten in bij verschillende soorten leveranciers. Naast het verkopen van deze producten leveren zij ook diensten. Dit zijn onderhoudsdiensten, 24uurs beveiliging, advisering etc. Deze diensten voeren zij uit middels processen. Enkele belangrijke processen binnen Complexi B.V. zijn het inkoop proces, het verkoop proces en het administratieve proces. Om deze processen goed te kunnen uitvoeren maakt Complexi B.V. gebruikt van verschillende applicaties en gegevens. Complexi B.V. heeft verschillende applicaties voor het verkopen van haar diensten. Zo beschikken zij over een verkoop applicatie voor particulieren en één voor zakelijke klanten. Het administratieve proces wordt ondersteund door diverse Baan modules en spreadsheets bij de medewerkers zelf. Het inkoop proces kent per product een aparte module. De applicaties zijn gemaakt in verschillende programmeertalen zoals Pascal, Cobol, C++ en Delphi. Elke applicatie is ontworpen middels een bepaalde ontwerpmethode. De twee belangrijkste zijn RUP en Yourdon. Sommige applicaties communiceren via middleware en andere applicaties communiceren rechtstreeks met het platform. Op het moment zijn er 2 soorten middleware aanwezig, namelijk .Net van Microsoft en J2EE van IBM. De platformen binnen Complexi B.V. bestaan uit verschillende soorten hardware bestaande uit pc's, mini's en mainframes met daarop draaiende systeemsoftware (Windows XP,98,95, Unix 8.0, AS/400, Linux 7.0). Deze hardware is verbonden met elkaar middels een netwerk. Het netwerk bestaat uit verschillende type switches, routers, hubs en bekabeling.

Appendix B: ORM-Modellen

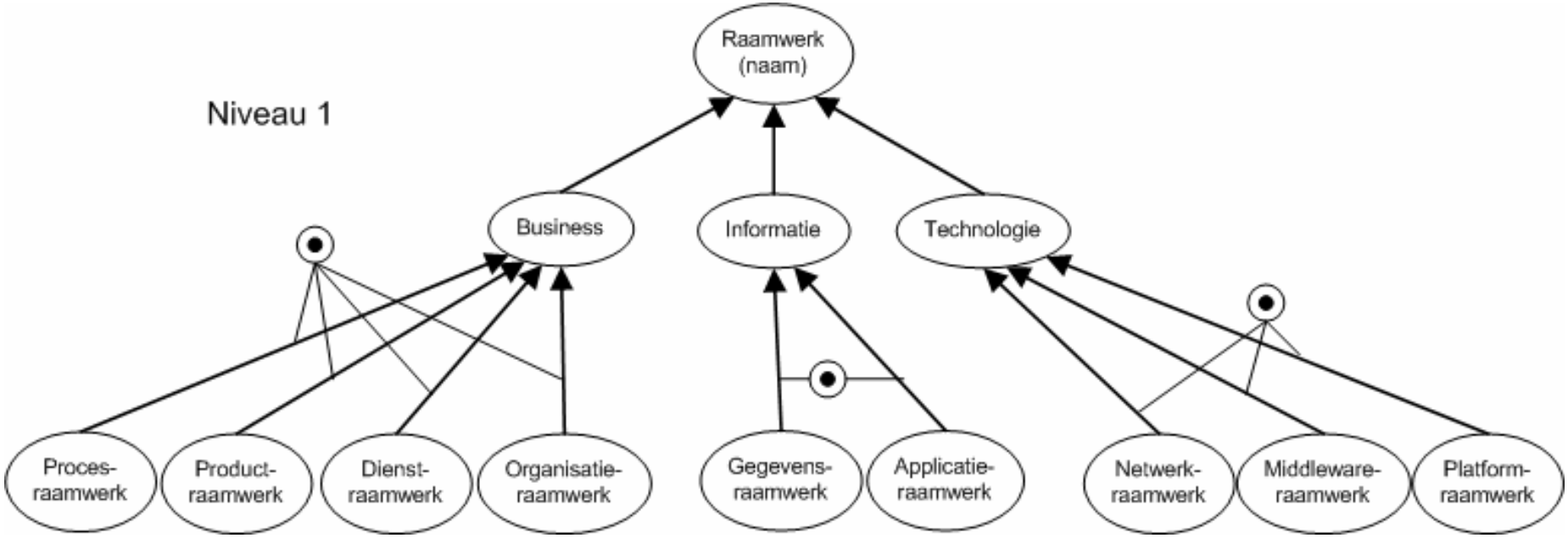
Het eerste niveau (Figuur 13 laat zien wat enterprise architectuur oplevert, in dit geval een raamwerk. Om een raamwerk te verkrijgen dienen er 9 views te worden ingevuld. Deze 9 views zijn verdeeld over de drie views van enterprise architectuur.

Het tweede niveau (Figuur 14) omschrijft hoe één van de 9 views ingevuld moet worden. Het woord domeinraamwerk is een algemene naam voor ieder van de 9 views. Om een domeinraamwerk in te vullen moet men de beschikking hebben over een domein. Binnen een domein gelden een aantal principes, daaronder vallen regels en richtlijnen. Aan de hand van het domein dienst zal dit worden uitgelegd. Het case bedrijf heeft het principe “Diensten en Producten zijn met elkaar verweven” hieronder valt de regel “Het leveren van maatwerk” en de richtlijn “kies uit vaste leveranciers”. Deze principes,regels en richtlijnen worden gemaakt door het domein dienst en moeten de onderliggende en bovenliggende principes niet tegen spreken, daarom heeft principe een relatie met zich zelf “is consistent met”. Uit die principes,regels en richtlijnen tezamen met de inhoud van het domein (niveau 4) worden modellen opgesteld om het geheel te visualiseren. Hierbij wordt gebruik gemaakt van een modelmethodiek.
Binnen het domein Dienst van het case bedrijf kunnen aan de hand van de principes en de gevoerde diensten modellen worden opgesteld met de relaties van de diensten met bijvoorbeeld de buitenwereld (ContextDiagram). Dit wordt gedaan volgens een methodiek (RUP). Dit tezamen moet een consistent geheel zijn.

Het derde niveau (Figuur 15) geeft de relaties weer tussen de verschillende views en laat zien dat ICT in dienst staat van de business. De views producten en diensten worden ondersteund door de processen. Een product kan ook onderdeel uitmaken van een of meerdere diensten. Om een proces goed te kunnen uitvoeren is er informatie nodig, dit komt enerzijds uit de applicaties en anderzijds uit de gegevens. De applicaties draaien op een platform en maken eventueel gebruik van middleware, die ook op een platform draait. Het platform maakt verbindingen door gebruik te maken van de netwerk infrastructuur.

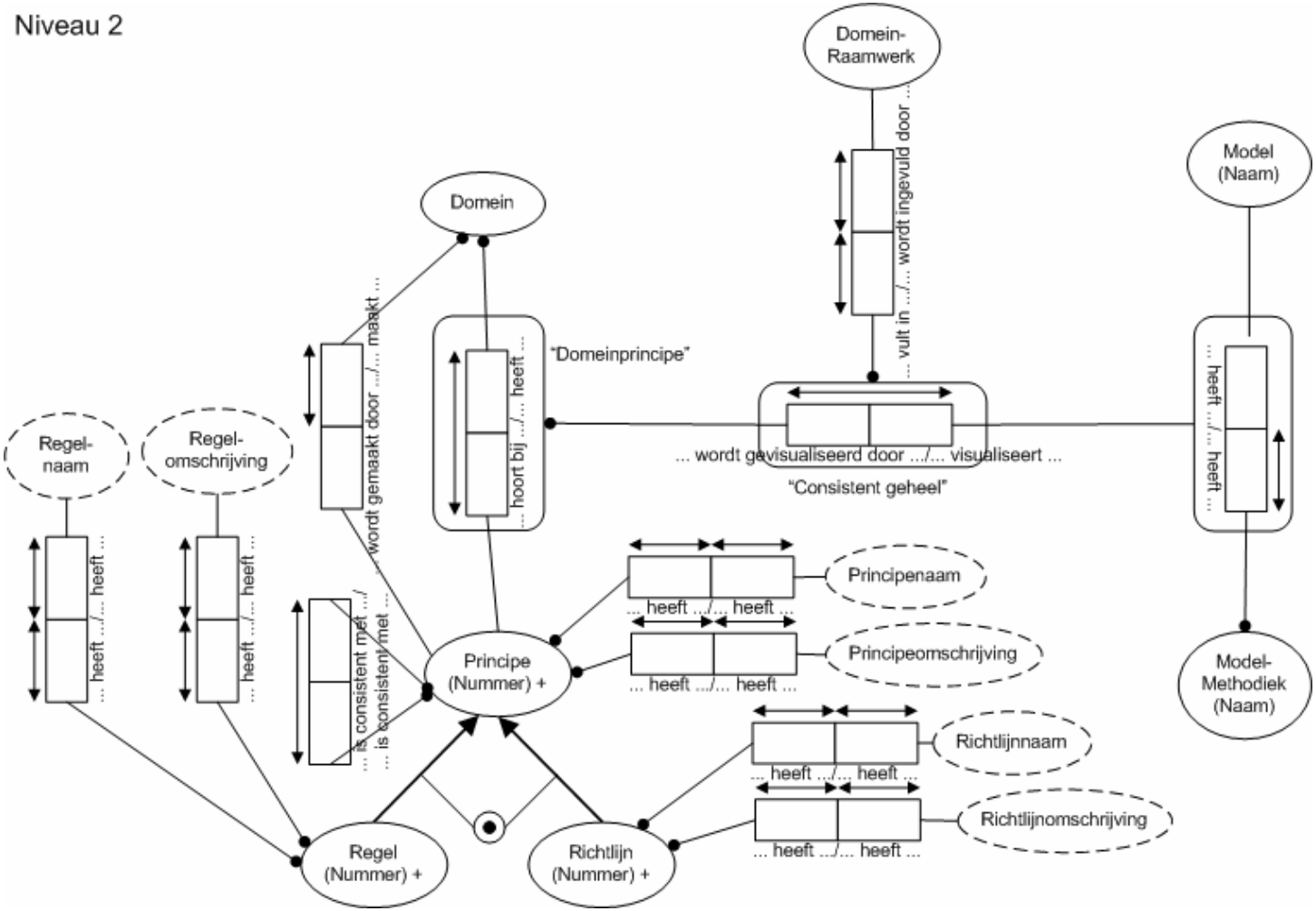
Binnen deze 9 views bevindt zich complexiteit, deze kan inzichtelijk gemaakt worden door de modellen op het vierde niveau (Figuur 16, Figuur 17, Figuur 18, Figuur 19, Figuur 20, Figuur 21, Figuur 22, Figuur 23). Uitleg over deze modellen wordt gegeven in paragraaf (3.5.1 tot en met paragraaf 3.5.3)

B1. ORM-Model Niveau 1



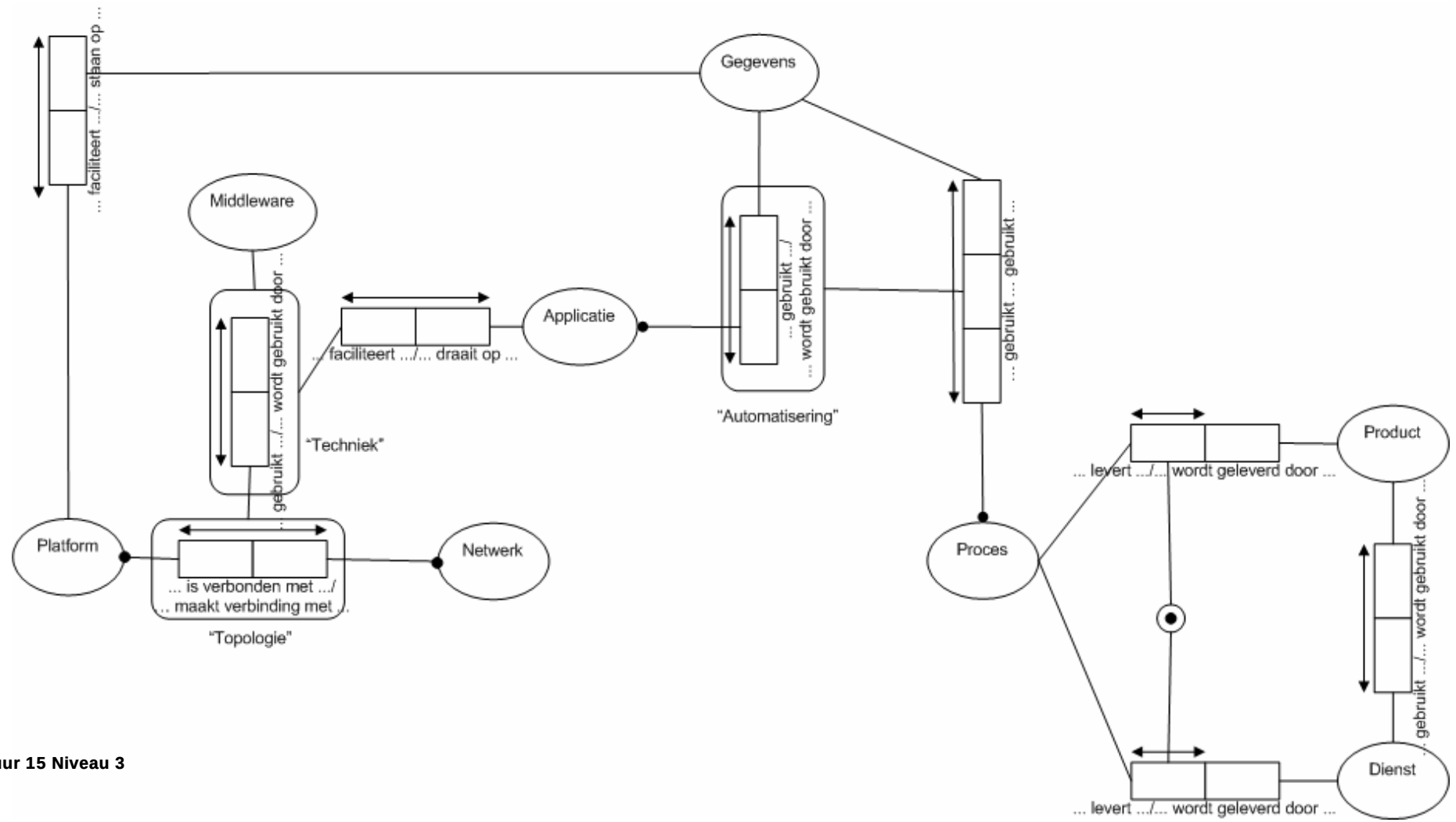
Figuur 13 Niveau 1

B2. ORM-Model Nveau 2



Figuur 14 Niveau 2

B3. ORM-Model Niveau 3

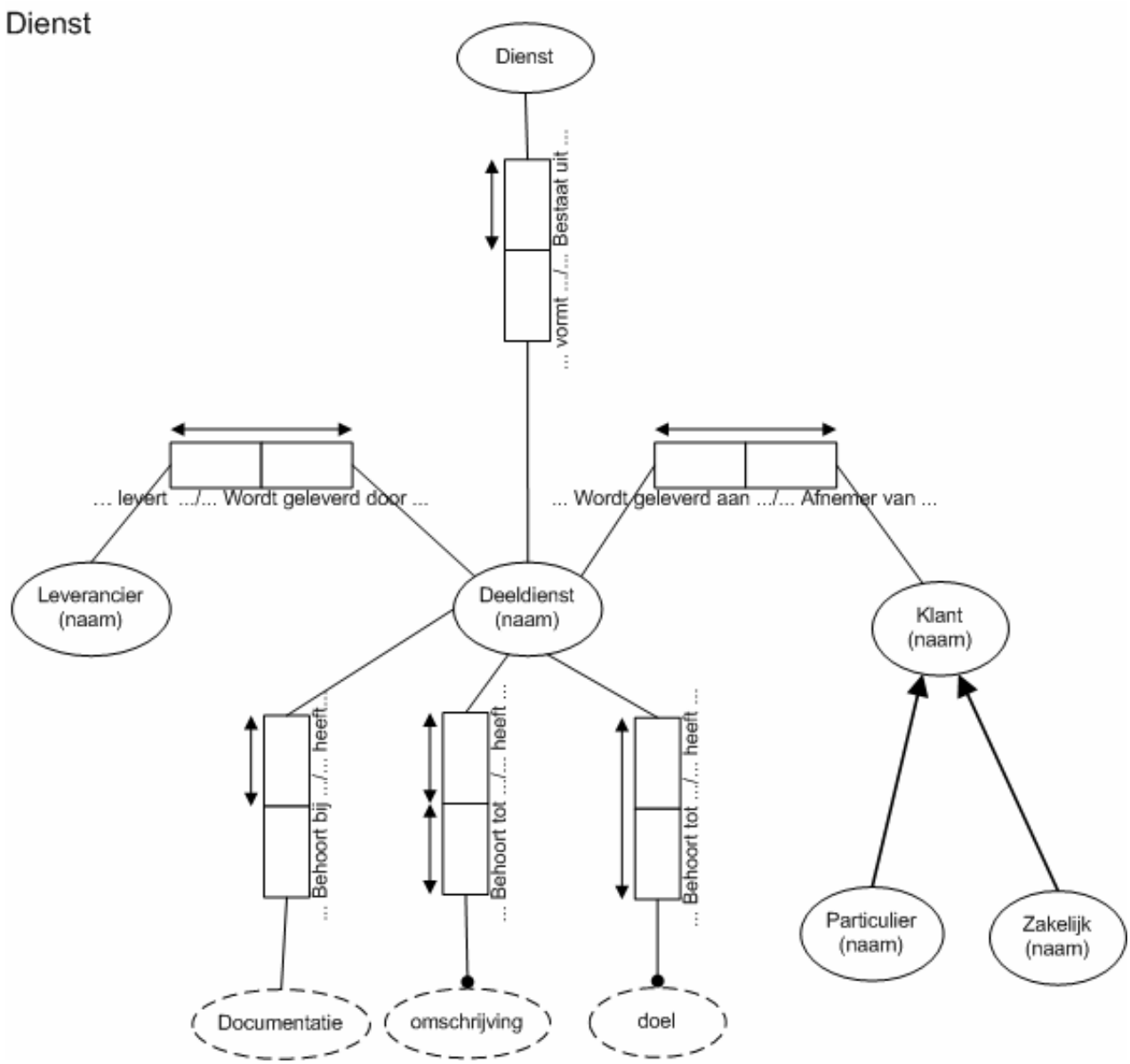


Figuur 15 Niveau 3

B4. ORM-Modellen Niveau 4

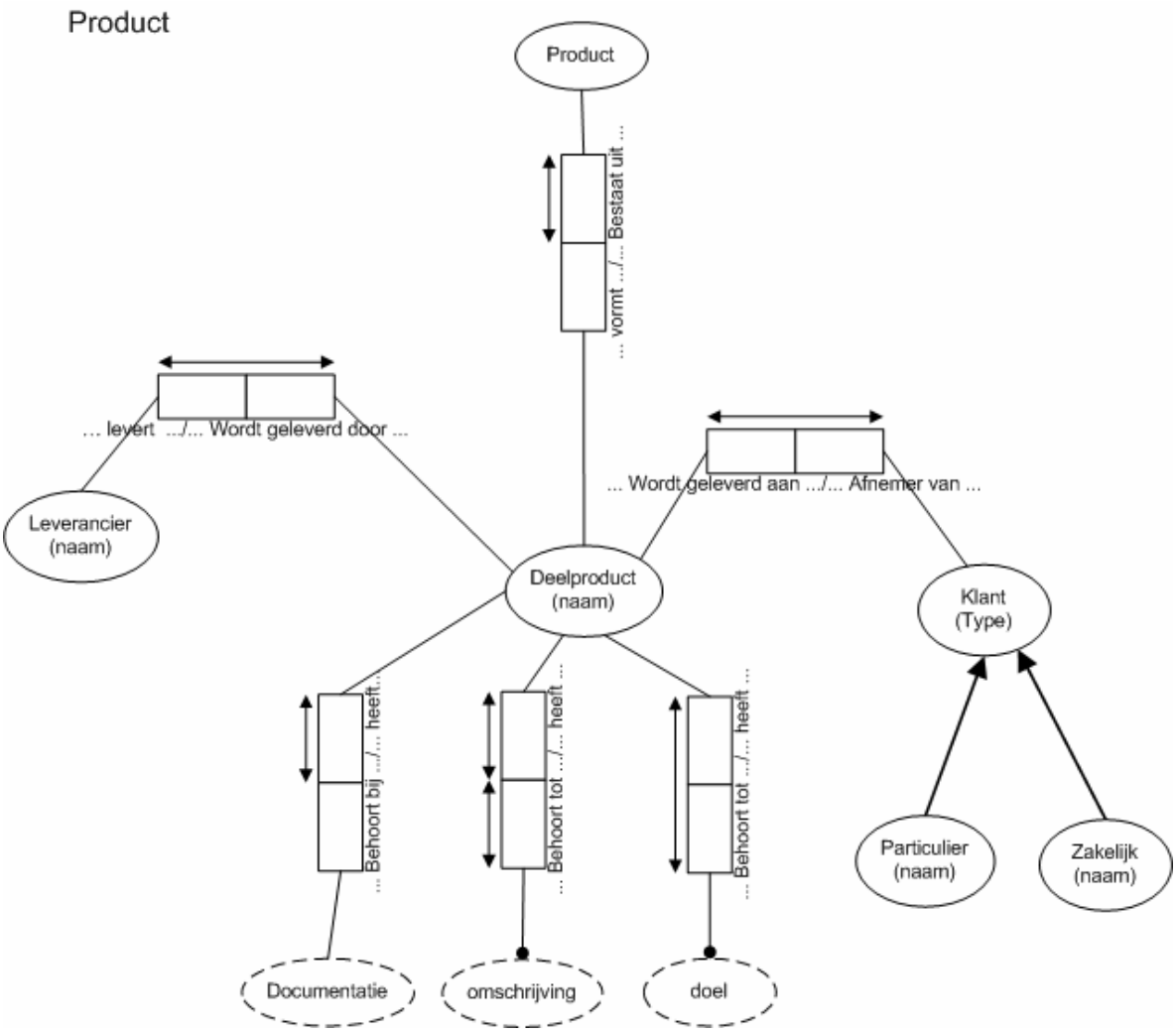
De objecten uit het ORM-model van het 3de niveau zijn in deze paragraaf verder uitgediept. De aspecten van complexiteit komen hierin naar voren.

B4.1 Dienstmodel



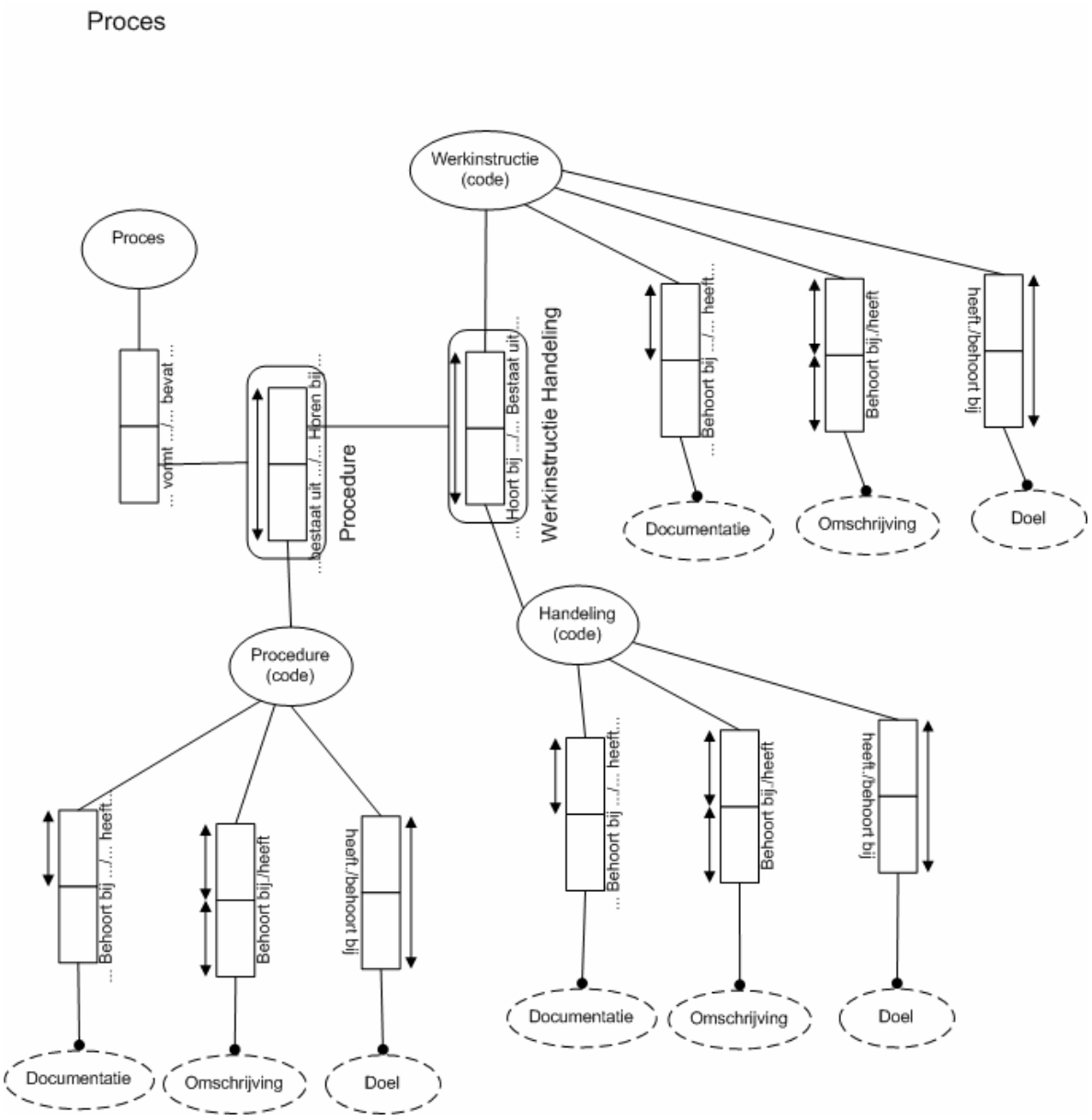
Figuur 16 niveau 4 Dienst

B4.2 Productmodel



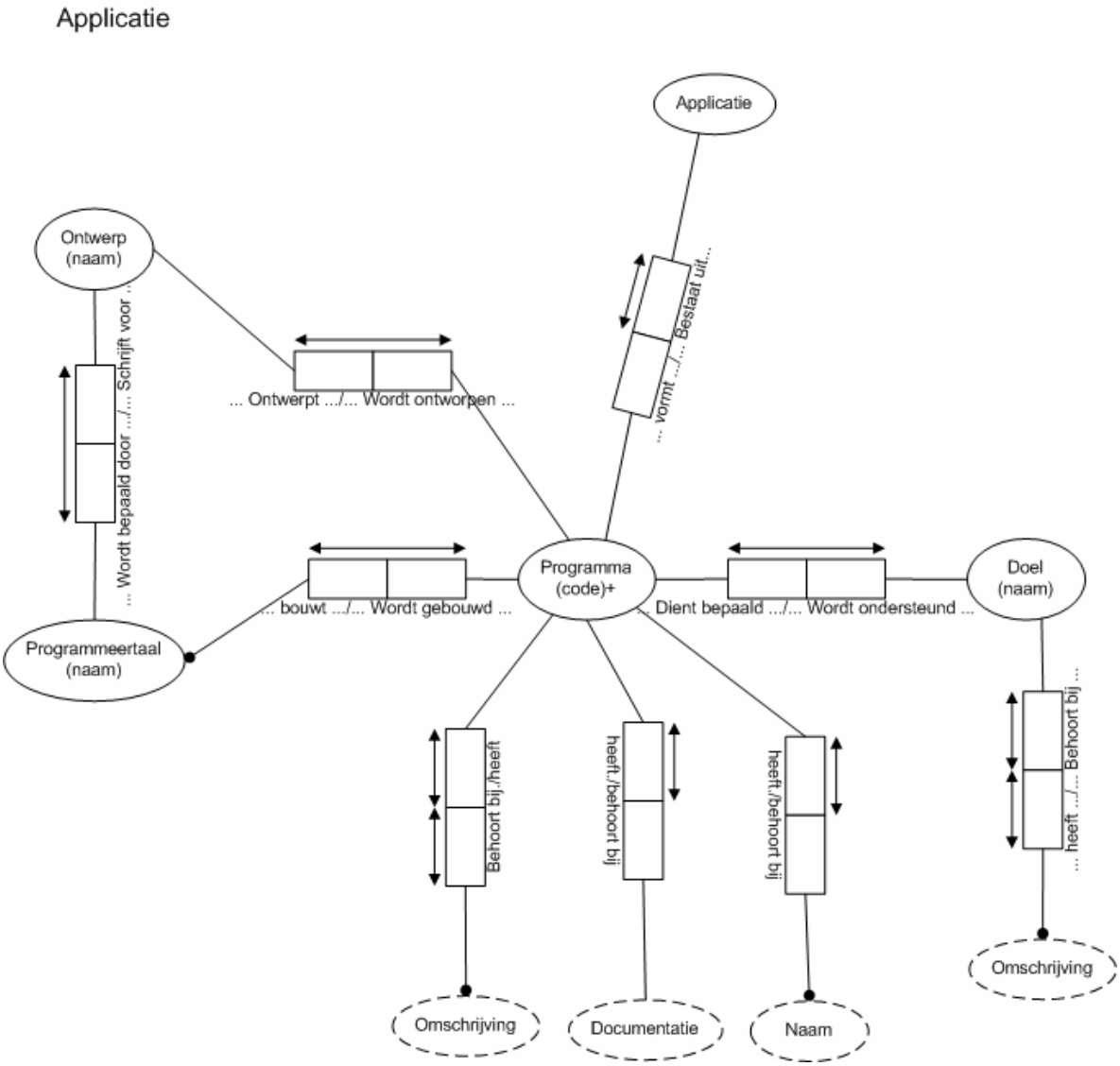
Figuur 17 niveau 4 Product

B4.3 Procesmodel



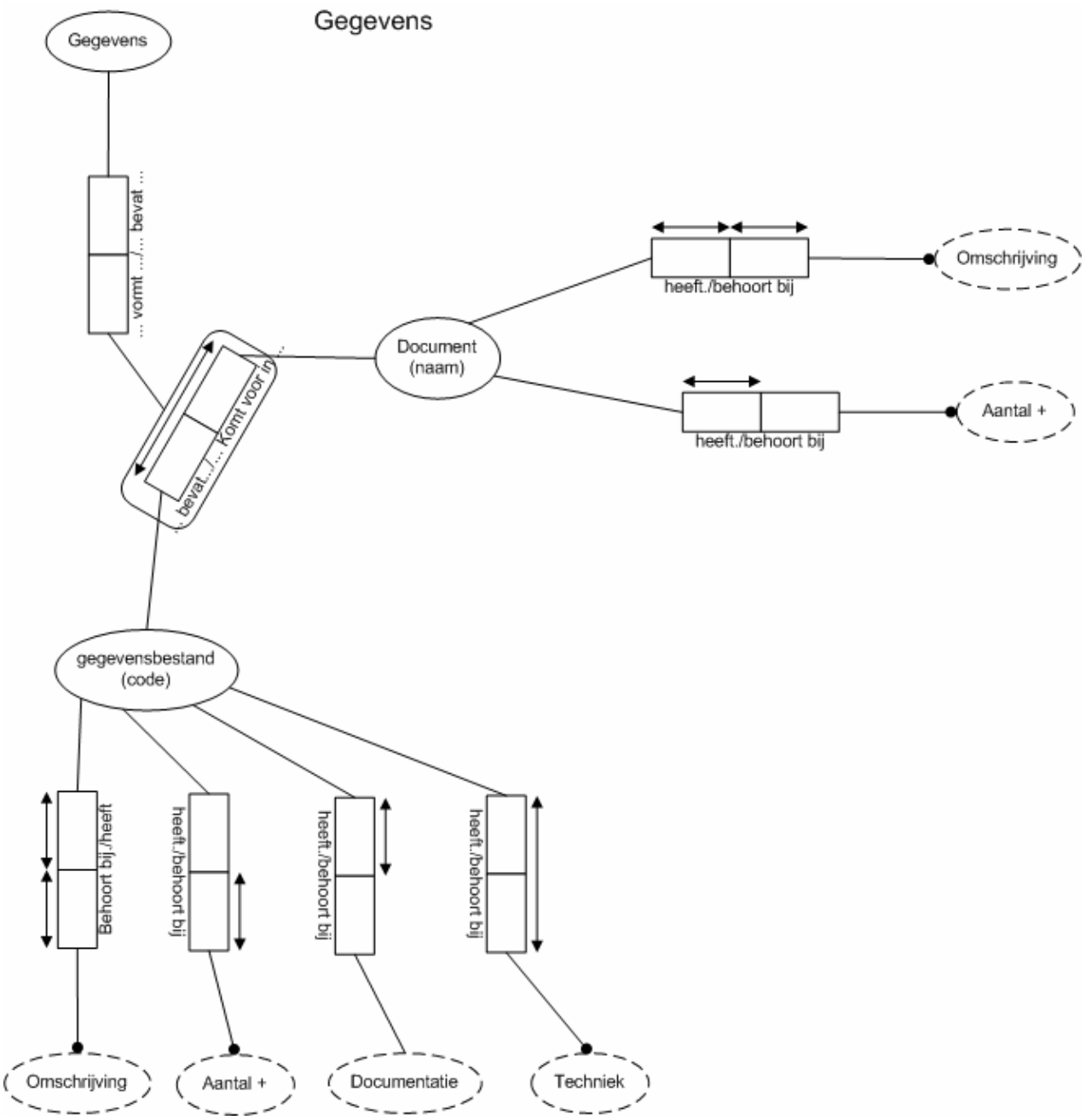
Figuur 18 niveau 4 Proces

B4.4 Applicatiemodel



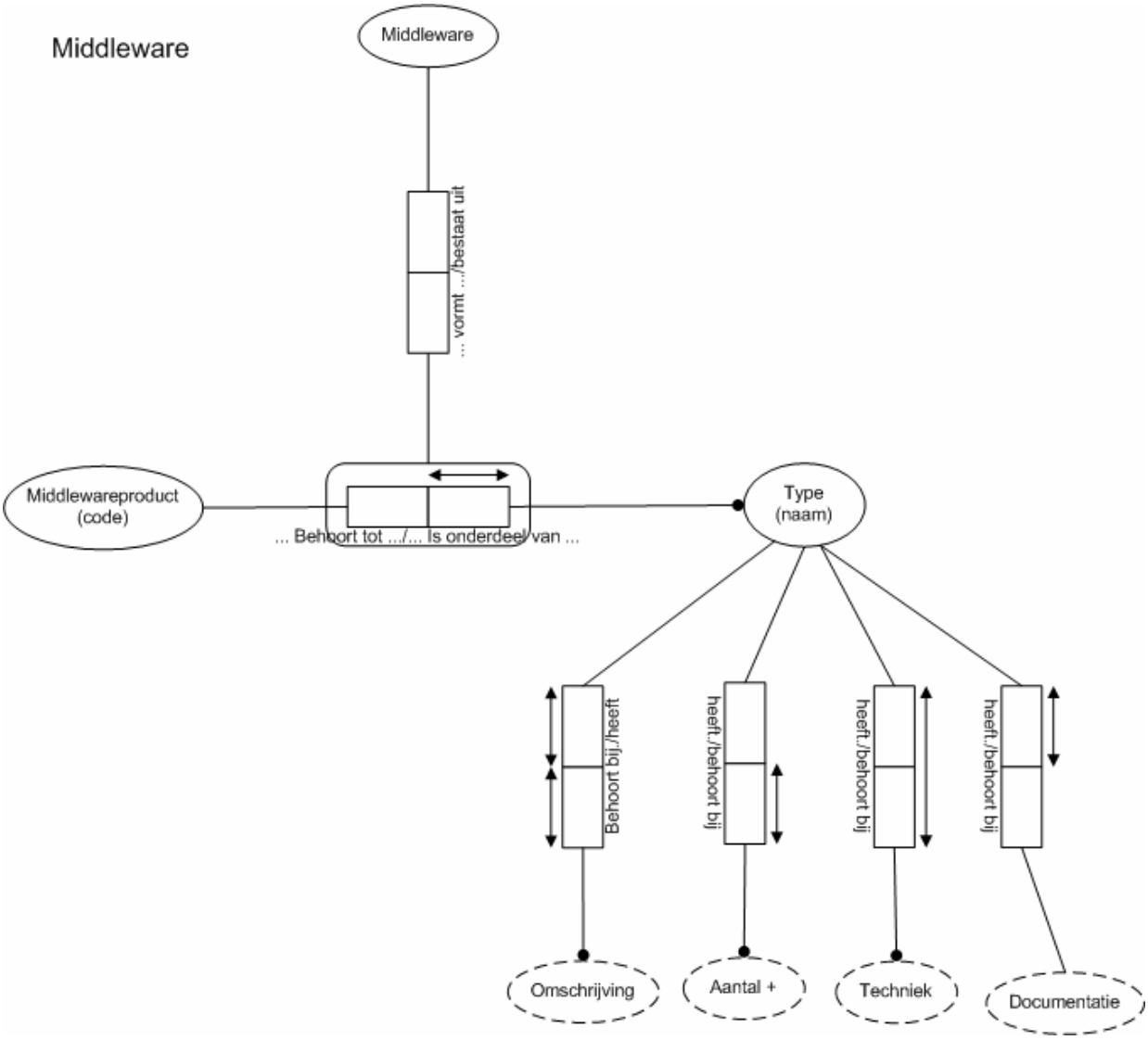
Figuur 19 niveau 4 applicatie

B4.5 Gegevensmodel



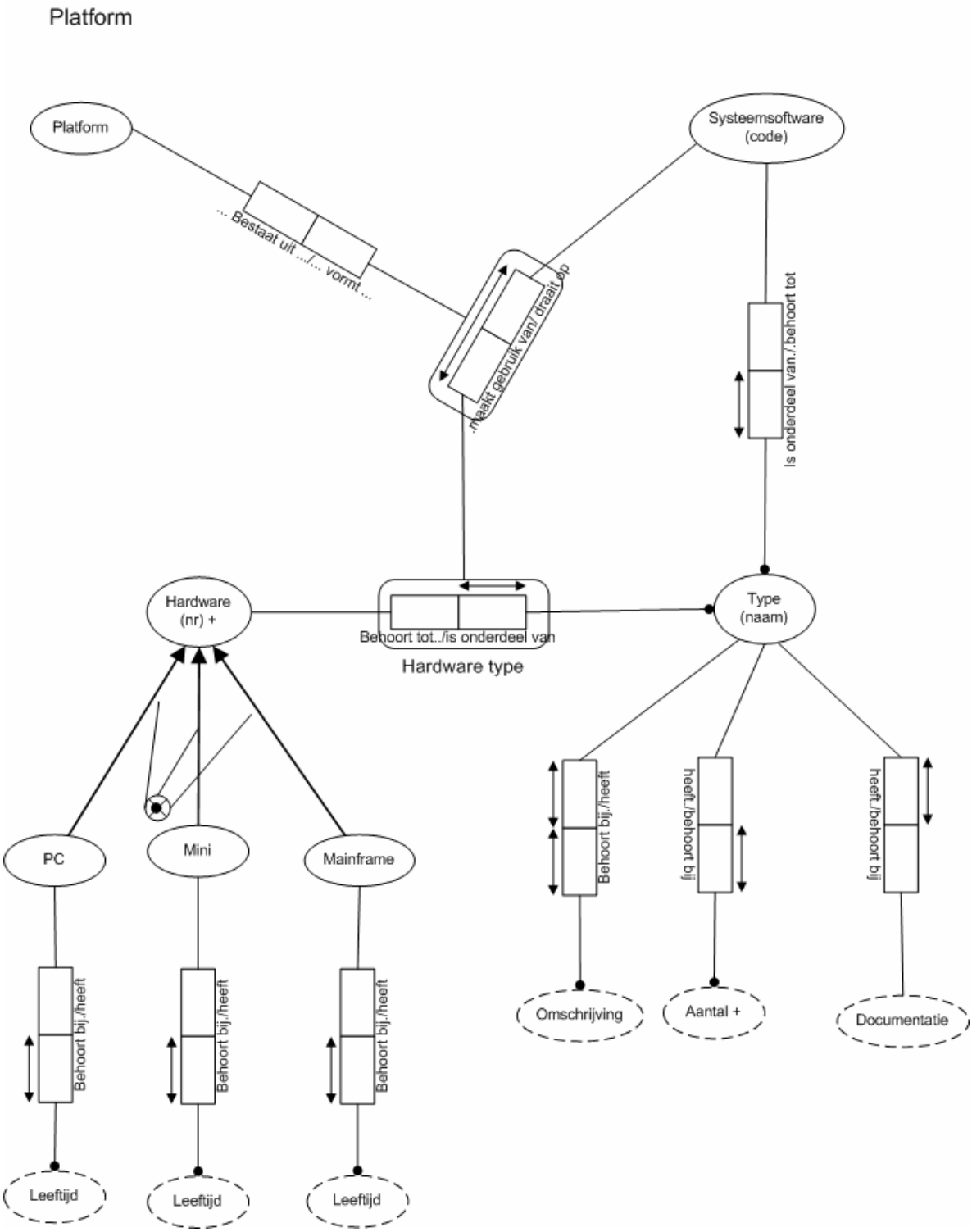
Figuur 20 niveau 4 Gegevens

B4.6 Middlewaremodel



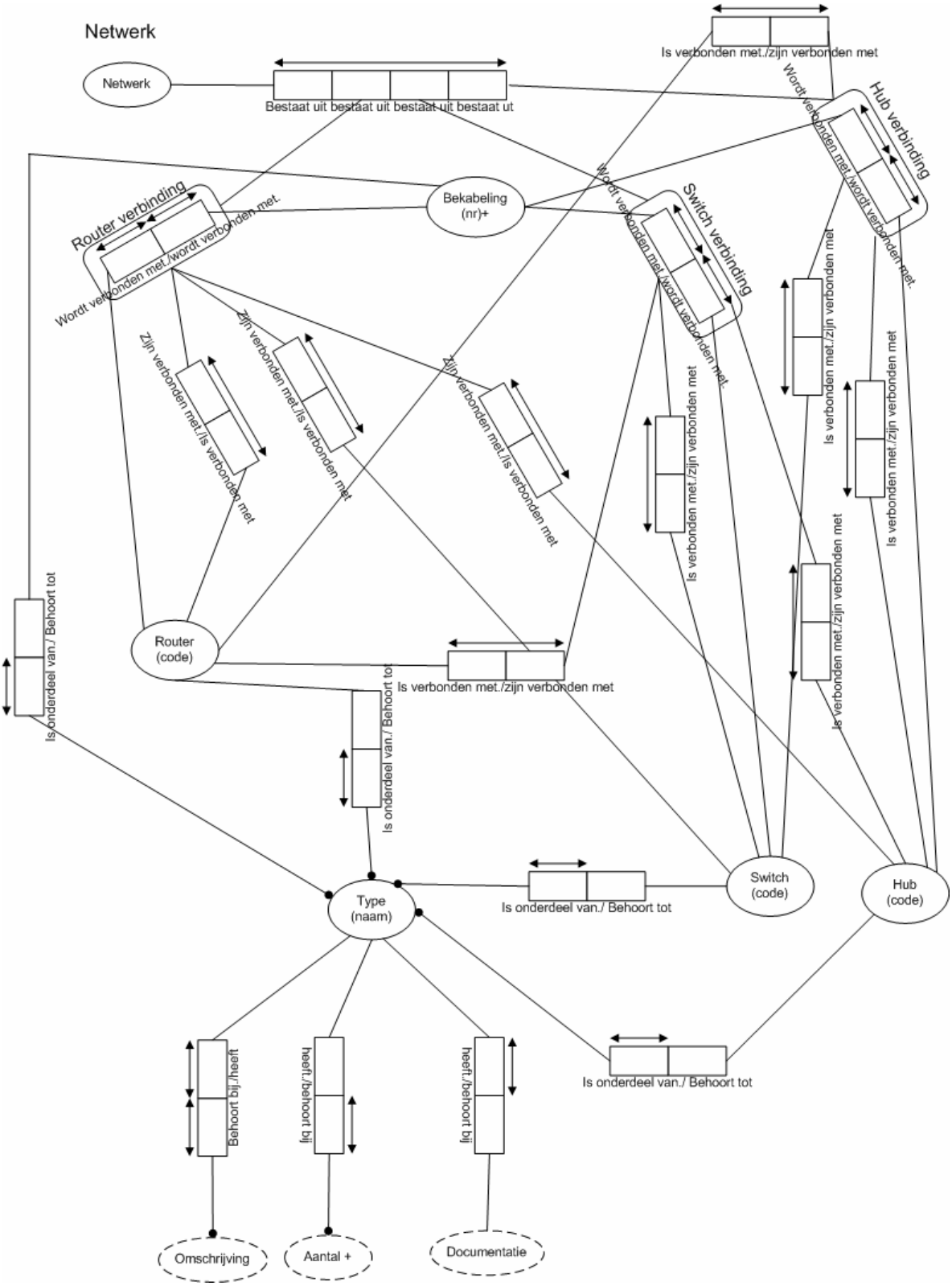
Figuur 21 niveau 4 Middleware

B4.7 Platformmodel



Figuur 22 niveau 4 Platform

B4.8 Netwerkmodel



Figuur 23 niveau 4 Netwerk

Appendix C: Uitwerking systeemontwerp middels logisch geformuleerde dependency-structuren

Deze Appendix beschouwt het onderzoeksgebied als een systeem. Door gebruik te maken van systeemontwerp middels logisch geformuleerde dependency-structuren wordt het onderzoeksgebied als een systeem omschreven. De achterliggende methodologie is te vinden in Appendix D.

C1. Hoofd-C en hoofd-A

Allereerst zal de hoofd C worden opgesteld. De hoofd C is hetgeen wat het systeem oplevert. In dit geval is dat een raamwerk waarin complexiteit valt af te lezen. De A's geven aan welke onderdelen de hoofd C daarvoor nodig heeft.

C raamwerk

A regelnaam, regelnummer, regelomschrijving, principenaam, principenummer, principeomschrijving, richtlijnnummer, richtlijnnaam, richtlijnomschrijving, modelmethodiek, model, leverancier, deeldienstnaam, deeldienstdocumentatie, deeldienstomschrijving, deeldienstdoel, klantparticulier, klantzakelijk, deelprodcutnaam, deelproductomschrijving, deelproductdoel, proceduredocumentatie, procedurecode, procedureomschrijving, proceduredoel, werkinstructiecode, werkinstructiedocumentatie, werkinstructieomschrijving, werkinstructiedoel, handelingcode, handelingdocumentatie, handelingomschrijving, handelingdoel, ontwerp, programmeertaal, programmacode, programmaomschrijving, programmadocumentatie, programmanaam, doelnaam, doelomschrijving, documentnaam, documentomschrijving, documentaantal, gegevensbestandcode, gegevensbestandomschrijving, gegevensbestandaantal, gegevensbestanddocumentatie, gegevensbestandtechniek, middlewareproduct, m_typenaam, m_typeomschrijving, m_typeaantal, m_typetechniek, m_typedocumentatie, systeemsoftwarecode, pc, mini, mainframe, p_typenaam, p_typeomschrijving, p_typeaantal, p_typedocumentatie, bekabeling, hub, switch, router, n_typenaam, n_typeomschrijving, n_typedocumentatie, n_typeaantal

C2. Uitwerking Niveau 1

Raamwerk

c1. raamwerk
c1. business, informatie, techniek, raamwerknaam

DEF raamwerk := business ∧ informatie ∧ techniek ∧ raamwerknaam
=> raamwerk

Business

c2. business
a2. productraamwerk, dienstraamwerk, procesraamwerk, organisatieraamwerk, businessnaam

DEF business := (productraamwerk ∨ dienstraamwerk) ∧ procesraamwerk ∧
Organisatieraamwerk ∧ businessnaam
=> business

Informatie

c3. informatie
a3. gegevensraamwerk, applicatieraamwerk,informatienaam

DEF informatie := gegevensraamwerk ∧ applicatieraamwerk ∧ informatienaam
=> informatie

Techniek

c4. techniek
a4. middlewareraamwerk, platformraamwerk, netwerkraamwerk,technieknaam

DEF technologie := middlewareraamwerk ∧ platformraamwerk ∧ netwerkraamwerk ∧
Technieknaam
=> techniek

C3. Uitwerking Niveau 2

Domeinraamwerk

c5. domeinraamwerk
a5. domeinraamwerknaam, model, modelmethodiek, principe, domein,

DEF domeinraamwerk := domeinraamwerknaam ∧ model ∧ modelmethodiek ∧ principe ∧ domein
=> domeinraamwerk

Principe

c6. principe
a6. domein, regel, richtlijn, principenummer, principenaam, principeomschrijving

DEF principe := domein ∧ principenummer ∧ principenaam ∧ principeomschrijving ∧ (regel ∨ richtlijn)
=> principe

Regel

c7. regel
a7. regelnummer, regelnaam, regelomschrijving

DEF regel := regelnummer ∧ regelnaam ∧ regelomschrijving
=> regel

Richtlijn

c8. richtlijn
a8. richtlijnnummer, richtlijnnaam, richtlijnomschrijving

DEF richtlijn := richtlijnnummer ∧ richtlijnnaam ∧ richtlijnomschrijving
=> richtlijn

C4. Uitwerking Niveau 3 + 4

Dienst

c9. dienst
a9. leverancier, deeldienst, klant, proces, product

DEF dienst := leverancier ∧ deeldienst ∧ klant ∧ proces ∨ product
=> dienst

Deeldienst

c10. Deeldienst
a10. omschrijving, doel, documentatie, deeldienstnaam

DEF Deeldienst := omschrijving ∧ doel ∧ documentatie ∧ deeldienstnaam
=> deeldienst

Product

c11. product
a11. leverancier, deelproduct, klant, proces

DEF product := leverancier ∧ deelproduct ∧ klant ∧ proces
=> product

Deelproduct

c12. deelproduct
a12. omschrijving, doel, documentatie, deelproductnaam

DEF Deelproduct := omschrijving ∧ doel ∧ documentatie ∧ deelproductnaam
=> deelproduct

Klant

c13. klant
a13. zakelijk, particulier

DEF Klant := zakelijk ∧ particulier
=> klant

Proces

c14. proces
a14. procedure, werkinstructie, handeling, applicatie, gegevens

DEF proces := procedure ∧ werkinstructie ∧ handeling (applicatie ∧ gegevens)
∨ gegevens
=> proces

Procedure

c15. procedure
a15. omschrijving, doel, documentatie, p_code

DEF Procedure := omschrijving ∧ doel ∧ documentatie ∧ p_code
=> procedure

Werkinstructie

c16. werkinstructie
a16. omschrijving, doel, documentatie, w_code

DEF Werkinstructie := omschrijving ∧ doel ∧ documentatie ∧ w_code
=> werkinstructie

Handeling

c17. handeling
a17. omschrijving, doel, documentatie, h_code

DEF Handeling := omschrijving ∧ doel ∧ documentatie ∧ h_code
=> handeling

Applicatie

c18. applicatie
a18. ontwerp, programmeertaal, programma, doel, gegevens, middleware, platform, netwerk, gegevens

DEF applicatie := ontwerp ∧ programmeertaal ∧ programma ∧ doel ∧
(platform ∧ netwerk ∧ gegevens) ∨ middleware
=> applicatie

Programma

c19. programma
a19. omschrijving, documentatie, naam, programma_code

DEF programma := omschrijving ∧ documentatie ∧ naam ∧ programma_code
=> applicatie

Doel

c20. doel
a20. omschrijving, doel_naam

DEF doel:= omschrijving ∧ doel_naam
=> doel

Gegevens

c21. gegevens
a21. gegevensbestand, document, platform

DEF gegevens := gegevensbestand ∧ document
=> gegevens

Gegevensbestand

c22. gegevensbestand
a22. omschrijving,aantal,documentatie,techniek,gegevensbestand_code

DEF gegevensbestand := omschrijving ∧ aantal ∧ documentatie ∧ techniek ∧
gegevensbestand_code
=> gegevensbestand

Document

c23. document
a23. aantal,omschrijving,document_naam

DEF document:= aantal ∧ omschrijving ∧ document_naam
=> document

Middleware

c24. middleware
a24. middlewareproduct,m_type

DEF middleware := middlewareproduct ∧ m_type
=> middleware

M_type

c25. m_type
a25. omschrijving,aantal,techniek,documentatie,m_type_naam

DEF m_type := omschrijving ∧ aantal ∧ techniek ∧ documentatie ∧ m_type_naam
=> m_type

Platform

c26. platform
a26. hardware,systeemsoftware

DEF platform := hardware ∧ systeemsoftware
=> platform

Hardware

c27. hardware
a27. pc,mini,mainframe,p_type,hardware_nr

DEF hardware :=pc ∧ mini ∧ mainframe ∧ p_type ∧ hardware_nr
=> hardware

Software

c28. software
a28. software_code,p_type

DEF software :=software_code ∧ p_type
=> software

P_type

c29. p_type
a29. omschrijving,aantal,documentatie,p_type_naam

DEF p_type:=omschrijving ∧ aantal ∧ documentatie ∧ p_type_naam
=> p_type

Netwerk

c30. netwerk
a30. router,hub,switch,bekabeling,n_type

DEF netwerk :=router ∧ hub ∧ switch ∧ bekabeling ∧ n_type
=> netwerk

N_type

c31. n_type
a32. omschrijving,aantal,documentatie,n_type_naam

DEF netwerk :=n_type_naam ∧ documentatie ∧ aantal ∧ omschrijving
=> n_type

Appendix: D. Methodologie

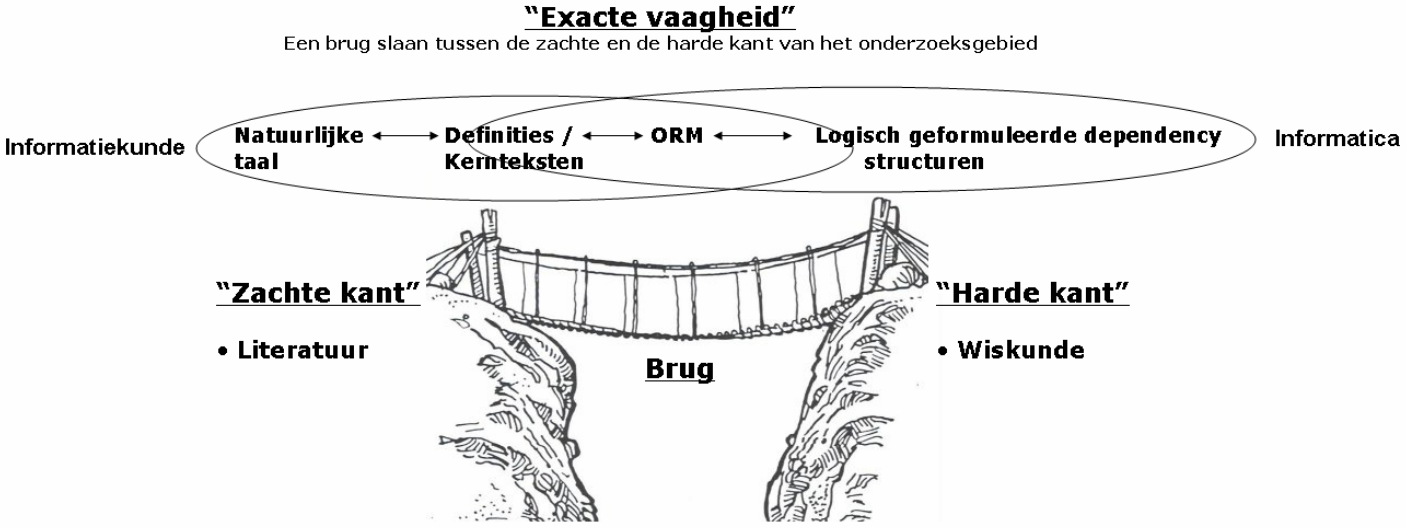
D1. Inleiding

Deze appendix behandelt de methodologie die gehanteerd is voor het in kaart brengen van het onderzoeksgebied middels de ORM-modellen volgens Terry Halpin [HALPIN], het wiskundig systeemontwerp middels logisch geformuleerde dependency-structuren volgens Wupper en Mader [WUPPER] en de koppeling daartussen.

Allereerst zal beschreven worden waarom het onderzoeksgebied gemodelleerd is. Vervolgens zullen de redenen voor de gebruikte technieken worden behandeld. Dan zal hierna dieper worden ingegaan op de gehanteerde ORM- en dependency-aanpak. Tevens wordt hierna de koppeling tussen ORM en logisch geformuleerde dependency-structuren besproken om tenslotte een conclusie te trekken.

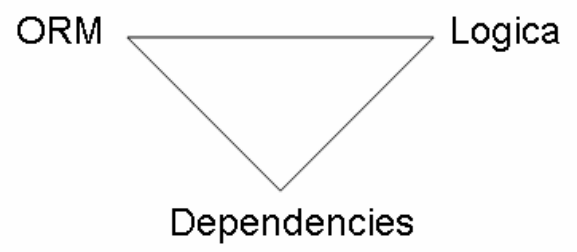
D1.1 Waarom onderzoeksgebied modelleren?

Tot op heden is er nog geen manier gevonden om de tekstuele en wiskundige weergave van een onderzoeksgebied goed met elkaar af te stemmen. Door gebruik te maken van methoden en technieken wordt er getracht een brug te slaan tussen de “zachte” en de “harde” kant van het onderzoeksgebied (zie figuur 24).



Figuur 24: Brug tussen de zachte en de harde kant onderzoeksgebied

De zachte (vage) kant dient exact gemaakt te worden om het onderzoeksgebied inzichtelijk te maken (ORM en logisch geformuleerde dependency-structuren) en om het onderzoeksgebied eenduidig vast te leggen (ORM en logisch geformuleerde dependency-structuren). De term “exacte vaagheid” is afkomstig van de inaugurele rede van dhr. prof. dr. Proper [PROPER]. De definities, de ORM en de logisch geformuleerde dependency-structuren vormen de brugcomponenten die tezamen het ravijn overbruggen tussen de “zachte” en de “harde” kant.



Figuur 25: Driehoeksrelatie ORM, Logica en Dependencies

In Figuur 25 is de driehoek getoond met de relaties tussen ORM, logica en dependencies. ORM geeft de relaties en rollen tussen objecten aan. Dependencies zijn specialisaties van relaties (ORM) uitgedrukt in simpele logische structuren (logica).

D1.2 Waarom natuurlijke taal?

Er is gekozen voor natuurlijke taal omdat alle gegevens die informatie bevat betreffende het onderzoeksgebied is weergegeven, vastgelegd of uitgewisseld in natuurlijke taal. Natuurlijke taal wordt in zijn algemeenheid nog steeds het meest gehanteerd. De kracht van natuurlijke taal is dat in principe alles door natuurlijke taal kan worden omschreven. Dit is tevens ook de zwakste kant aangezien natuurlijke taal kan leiden tot uiteenlopende notaties, foute syntax, homoniemen, geen eenduidigheid van begrippen en incorrectheid van een redenering. De informatie uit de natuurlijke taal vormen de basis voor de definities en kernteksten.

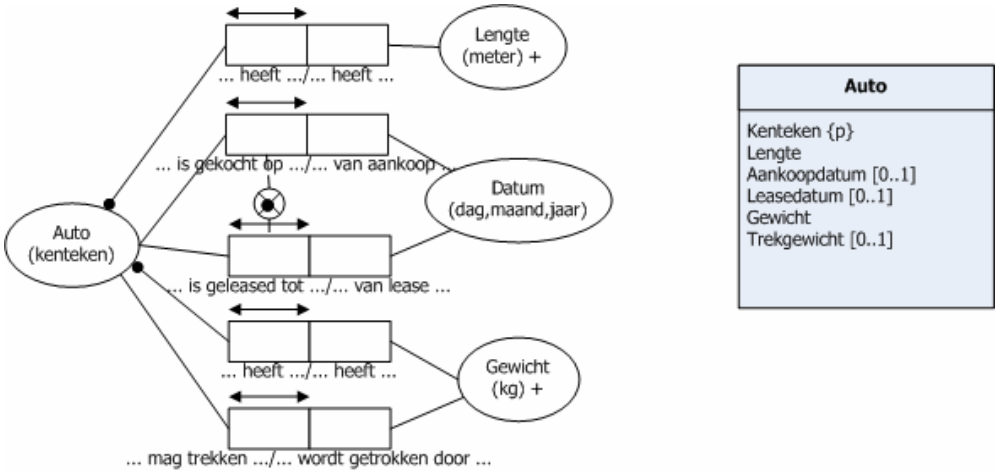
D1.3 Waarom definities en kernteksten?

Definities en kernteksten worden gebruikt om de informatie uit het onderzoeksgebied af te bakenen en te concretiseren. Een definitie is een omschrijving van de kenmerken en de betekenis van een begrip of een woord. Een kerntekst geeft een omschrijving van relevante informatie uit het onderzoeksgebied. De definities en kernteksten kunnen vervolgens vertaald worden richting modellen.

D1.4 Waarom ORM?

Het onderzoeksgebied is tot op heden nog niet in kaart gebracht middels een modelleertechniek. Om dit te kunnen doen is er een aangepaste methodiek nodig. Er zijn verschillende modelleertechnieken beschikbaar zoals bijvoorbeeld Unified Modeling Language (UML), Entity Relationship Diagram (ERD) en Object Role Modeling (ORM).

Er is gekozen voor ORM om drie redenen. De eerste reden is omdat, voor het beantwoorden van de hoofd- en deelvragen van het onderzoek, het onderzoeksgebied opgedeeld moet worden in objecten en hun onderlinge relaties. ORM is hier geschikter voor dan UML omdat bij bijvoorbeeld UML geen gebruik gemaakt kan worden van rollen. Voor het onderzoek is het belangrijk om te weten welke rollen er zijn tussen de objecten om hiermee de betekenis vast te leggen. ORM is tevens geschikter dan ERD omdat bij bijvoorbeeld ERD geen mogelijkheid bestaat tot het gebruiken van populatietabellen. Populatietabellen zijn nuttig omdat deze inzicht verschaffen in de relaties tussen de instanties van objecten.



Figuur 26: Vergelijking ORM en UML; UML toont minder detail [HALPIN]

In Figuur 26 is een voorbeeld gegeven van het modelleren van een auto middels ORM en UML. ORM toont de rollen, objecten en relaties op een gedetailleerde wijze. Het voorbeeld in UML laat de objecten en relaties zien, maar de constraints kunnen in UML niet weergegeven worden. In ORM kan bijvoorbeeld worden weergegeven dat een auto óf gekocht is, óf geleasd; en niet beide.

Een tweede reden voor het gebruiken van ORM is dat de onderzoeksgroep Informatie en Kennissystemen (IRIS), ORM gekozen heeft als standaard modelleertechniek. De afstudeerbegeleider heeft ORM daarom geadviseerd om toe te passen bij het onderzoek.

Als laatste reden is er bij de afstudeerders ten opzichte van UML en ERD meer kennis en ervaring aanwezig betreffende het gebruik van ORM.

D1.5 Waarom systeemontwerp middels logisch geformuleerde dependency-structuren?

Het onderzoeksgebied is te beschouwen als een systeem. Systeemontwerp middels logisch geformuleerde dependency-structuren biedt een mogelijkheid om het onderzoeksgebied als systeem op te delen. Het systeem bestaat uit onderdelen die tezamen een bepaald doel dienen te verwezenlijken. Het resultaat dat opgeleverd wordt door het systeem draagt bij aan de beantwoording van de onderzoeksvraag. Door middel van logisch geformuleerde dependency-structuren zijn gevolgtrekkingen te onderkennen. Deze gevolgtrekkingen komen veel voor binnen het onderzoeksgebied.

Een voorbeeld uit het onderzoek is dat *als* er een principenummer, een principenaam en een principeomschrijving is, *dan* is er sprake van een principe. Deze *als-dan-constructie* is een gevolgtrekking.

Systeemontwerp middels logisch geformuleerde dependency-structuren is daarnaast een manier om het onderzoeksgebied formeel vast te leggen. Ook kunnen hiermee de hiërarchie en afhankelijkheden binnen het systeem worden getoond.

D2. ORM-aanpak

Nadat er gekozen is voor ORM dient ORM onder de loep te worden genomen om te bekijken welke onderdelen van nut zijn voor het in kaart brengen en visualiseren van het onderzoeksgebied. Daarnaast dient een stappenplan opgesteld te worden dat inzicht verschaft in hoe het probleemgebied vertaald moet worden naar een ORM-model.

D2.1 Stap 1: Literatuurstudie

Alvorens er gestart kan worden met het modelleren dient het doel bepaald te worden. Het doel is vastgelegd in de hoofd- en deelvragen van het onderzoek. Vanuit deze hoofd- en deelvragen moet het onderzoeksgebied opgedeeld worden in domeinen. Een voorbeeld van een domein is bijvoorbeeld: “wat is enterprise architectuur?”. Van dit voorbeelddomein dient een ORM-model opgesteld te worden.

Nu het duidelijk is welke domeinen gemodelleerd moeten worden dient er relevant materiaal verzameld te worden. Dit materiaal kan uit de literatuur (artikelen, boeken, internet etcetera) en broneigenaren worden gehaald. Belangrijke stukken tekst en definities betreffende het domein dienen nu uit het materiaal te worden gefilterd. Deze kernteksten leveren de invoer voor de eerste ORM-schets.

Een voorbeeld van een kerntekst uit het onderzoek is de definitie van enterprise architectuur van Sogeti:

Architectuur is het consistente geheel aan principes en modellen dat richting geeft aan ontwerp en realisatie van processen, organisatorische inrichting, informatievoorziening en technische infrastructuur van een organisatie.

D2.2 Stap 2: Objecten vaststellen

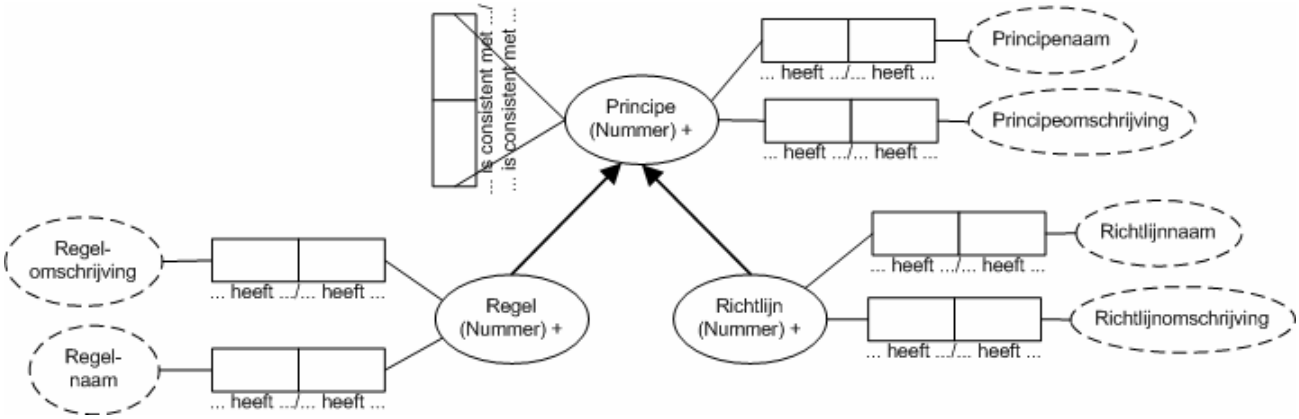
kernteksten dienen in deze stap vertaald te worden naar objecten en attributen. Uit de kernteksten moeten kernwoorden worden vastgesteld. Vaak zijn deze kernwoorden zelfstandige naamwoorden. Echter moet hierbij goed nagedacht worden over welke woorden wel en niet relevant zijn voor het onderzoek. Als de objecten zijn vastgesteld moeten de eigenschappen (de attributen) per object bepaald worden.

Bijvoorbeeld de kernwoorden uit de gekozen definitie van enterprise architectuur zijn: principe, model, proces, organisatie, informatievoorziening en technische infrastructuur. Vervolgens moeten per object de attributen worden bepaald. Voor principe zijn dat bijvoorbeeld: principenummer, principenaam en principeomschrijving.

D2.3 Stap 3: Relaties en rollen bepalen

Hoe een object wordt gevormd is niet altijd in de definitie te vinden. Daarom dient ook de kerntekst, gerelateerd aan het betreffende object, geraadpleegd te worden. Een voorbeeld uit het onderzoek is dat het object principe bestaat uit regels en richtlijnen. Van de objecten regel en richtlijn dienen ook attributen te worden opgesteld.

In deze derde stap dienen de nuttige relaties voor het onderzoek tussen de objecten te worden vastgesteld. Tussen de objecten bestaan verschillende soorten relaties: unaire, binaire, ternaire en quartaire relaties. Tevens bestaan er subtype-relaties zoals bij de objecten principe, regel en richtlijn. De objecten principe, regel en richtlijn en hun bijbehorende attributen worden getoond in figuur 27.

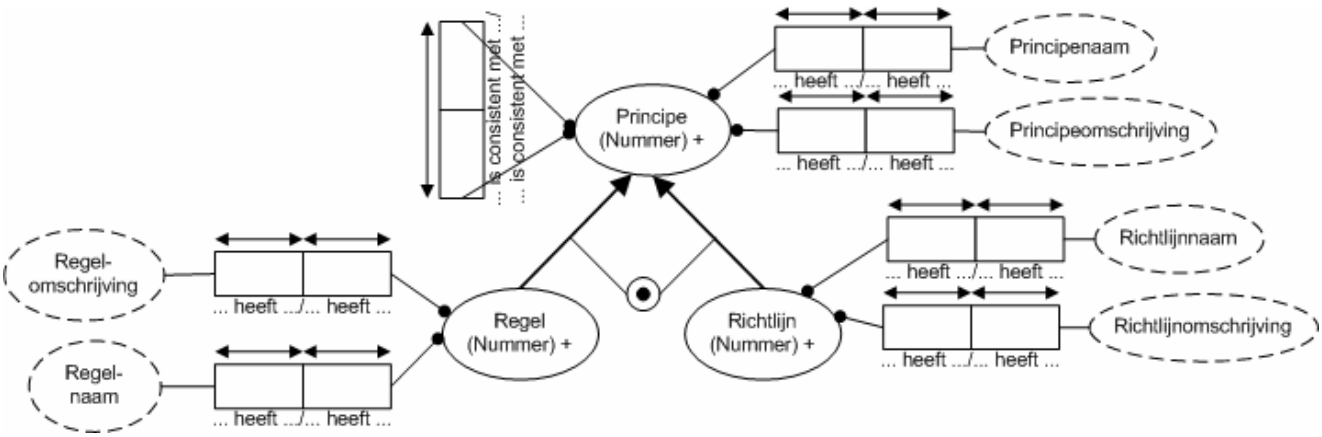


Figuur 27: Voorbeeld relaties en rollen tussen de objecten principe, regel en richtlijn

Ook moeten er rollen worden benoemd. Tussen objecten en attributen kan de rol “heeft” worden gebruikt. De rollen tussen objecten onderling hebben vaak een betekenisvolle benaming. In figuur 27 is bijvoorbeeld te zien dat principe 1 consistent dient te zijn met principe 2.

D2.4 Stap 4: Constraints toevoegen

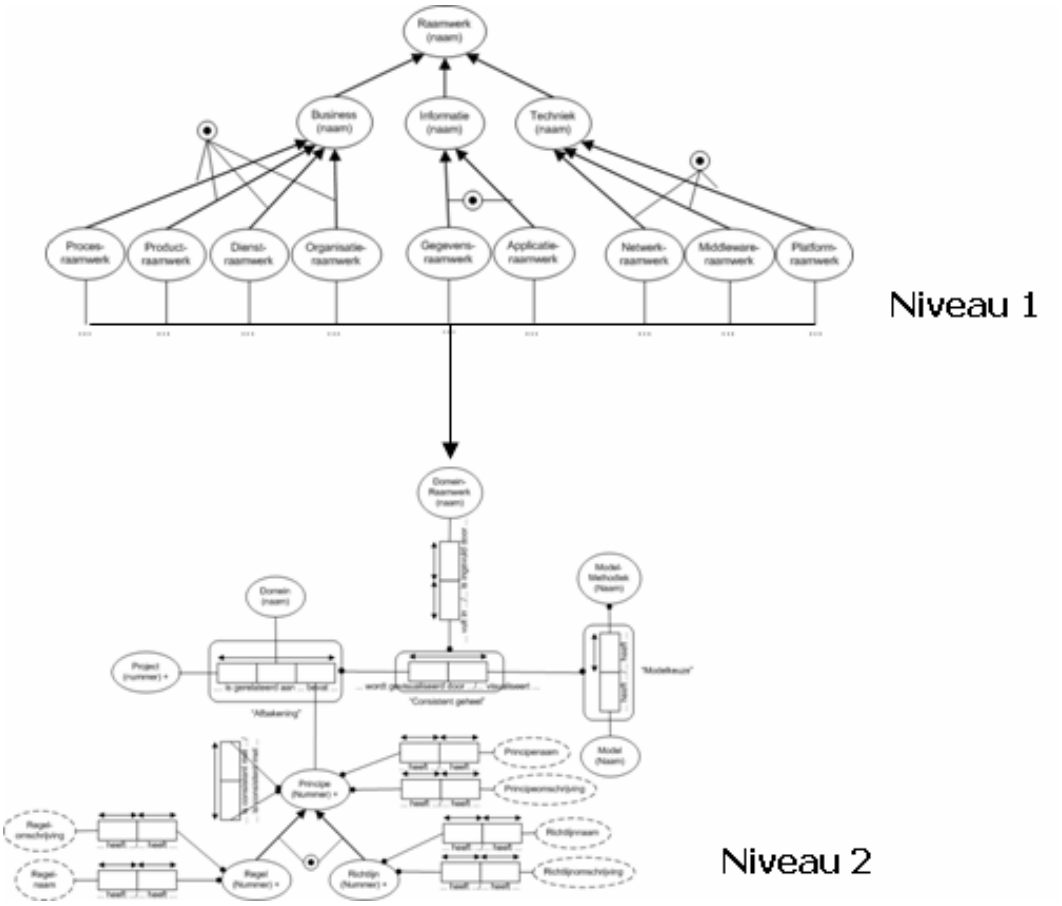
In deze stap dienen de constraints in het model aangebracht te worden. Er zijn [1..n]-, [1..1]- en [n..n]-relaties te onderkennen. Ook moet aangegeven worden welke objecten en attributen verplicht en niet verplicht zijn. Figuur 28 toont het voorbeeld uit het onderzoek betreffende de objecten principe, regel en richtlijn in combinatie met de constraints.



Figuur 28: Voorbeeld constraints tussen de objecten principe, regel en richtlijn

D2.5 Stap 5: verfijnen ORM-model

In stap 5 moet de huidige versie van het ORM-model worden verfijnd. Tekentechnisch kunnen de ORM-modellen leesbaarder gemaakt worden door het toepassen van bijvoorbeeld enrollments. Door middel van enrollments bundel je relaties. Hierdoor beperk je het aantal relaties waardoor de complexiteit van het model afneemt en het overzicht bewaard blijft.



Figuur 29: Opdeling enterprise architectuur in niveaus

Bij bijvoorbeeld het modelleren van enterprise architectuur is ook getracht het overzicht te bewaren. Dit is gedaan door enterprise architectuur op te delen in niveaus. Bijvoorbeeld de businesscomponent in het enterprise architectuur raamwerk bestaat uit vier onderdelen: product, dienst, organisatie en proces. Deze vier onderdelen worden in een niveau dieper generiek behandeld om tot een ingevuld business-component-raamwerk te komen.

D3. Systeemontwerp middels logisch geformuleerde dependency-structuren

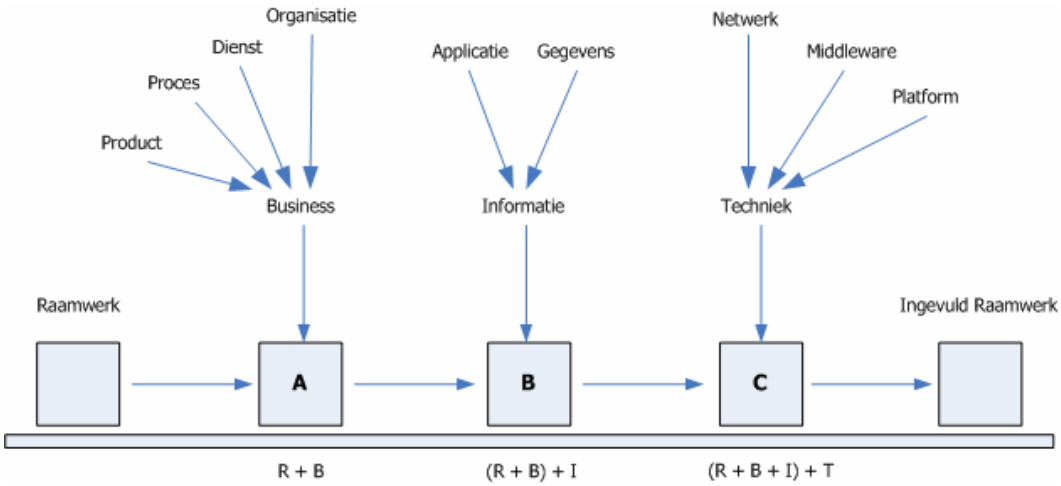
Nadat er gekozen is voor systeemontwerp middels logisch geformuleerde dependency-structuren dient een stappenplan opgesteld te worden dat inzicht verschaft in hoe het probleemgebied vertaald moet worden van een ORM-model naar logisch geformuleerde dependency-structuren. Voor de achterliggende theorie die in dit stappenplan gebruikt is wordt verwezen naar Wupper en Mader [WUPPER].

D3.1 Stap 1: doel bepalen

Het onderzoeksgebied en de ORM-modellen dienen nu als systeem beschouwd te worden. Met de eerste stap wordt bepaald wat het systeem oplevert; wat het doel is van het systeem. In deze dependency-methode wordt dat eindresultaat gekenmerkt als de “hoofd-C” (commitment). Bijvoorbeeld een “ingevuld raamwerk” zou een eindresultaat kunnen zijn van het onderzoeksgebied. Nadat de hoofd-C is bepaald moet het ORM-model grondig onder de loep worden genomen om te bepalen welke objecten als deelsystemen moeten dienen om tot dit eindresultaat te komen.

D3.2 Stap 2: maken systeemschets

Om tot het eindresultaat te komen verkeert het systeem in verschillende toestanden. Door het maken van een systeemschets wordt het systeem en zijn toestanden gevisualiseerd op een bepaald abstractieniveau. Per abstractieniveau kan een aparte schets gemaakt worden. In Figuur 30 is de systeemschets te zien betreffende niveau 1 uit Figuur 29



Figuur 30: Enterprise architectuur middels logisch geformuleerde dependency-structuren

Figuur 30 geeft een voorbeeld van hoe een systeemschets middels logisch geformuleerde dependency-structuren opgesteld zou kunnen worden. Echter is deze figuur niet prescriptief. Tevens is er voor gekozen om de systeemschets lineair te maken in plaats van iteratief, omdat de systeemschets anders onnodig moeilijk en onoverzichtelijk is.

Een raamwerk verkeert op een bepaald tijdstip in een lege toestand, de begintoestand op tijdstip 0. Vervolgens wordt in toestand A op tijdstip 0 de businesscomponent ingevuld, bestaande uit vier losse deelsystemen; product, proces, dienst en organisatie. Na tijdstip 0 + d1 (een bepaalde duratie) verandert de toestand van het raamwerk naar toestand B. In toestand B wordt de informatiecomponent ingevuld bestaande uit de deelsystemen applicatie en gegevens. Na tijdstip 0 + d1 + d2 verandert de toestand van het raamwerk naar toestand C. De techniekcomponent wordt vervolgens ingevuld. De techniekcomponent bestaat uit de deelsystemen netwerk, middleware en platform. Na tijdstip 0 + d1 + d2 + d3 is het raamwerk ingevuld.

D3.3 Stap 3: opstellen definities

In de derde stap dienen per systeemonderdeel definities opgesteld te worden. In figuur 30 is te zien dat bijvoorbeeld een ingevuld raamwerk wordt gevormd vanuit het business-, informatie- en techniekraamwerk.

Een voorbeeld van een definitie volgens logisch geformuleerde dependency-structuren is als volgt:

Ingevuld Raamwerk:

c1. ingevuld raamwerk
a1. business, informatie, techniek

DEF ingevuld raamwerk := business $\wedge$ informatie $\wedge$ techniek
=> ingevuld raamwerk

Business:

c2. business
a2. product, dienst, proces, organisatie

DEF business := (product $\vee$ dienst) $\wedge$ proces $\wedge$ organisatie
=> business

... etcetera

D3.4 Stap 4: bewijzen middels natuurlijke deductie

De opgestelde predikaten kunnen door middel van natuurlijke deductieregels [BENTHEM] worden bewezen. Bij de poging, een stelling te bewijzen, kan duidelijk worden dat assumptions (aannames) vergeten zijn. Tevens is het bewijzen een goed middel om onuitgesproken veronderstellingen boven water te krijgen en kun je door middel van een bewijs kijken of stellingen daadwerkelijk kloppen.

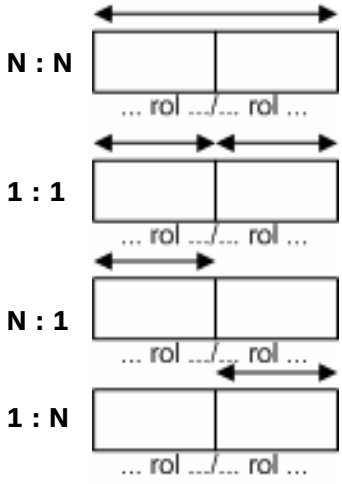
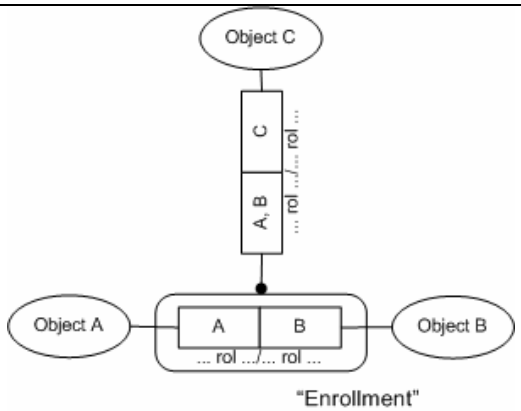
Deze vijfde stap is niet uitgevoerd binnen het onderzoek wegens tijdgebrek.

D4. Koppeling tussen ORM en logisch geformuleerde dependency-structuren

Op dit moment is de strategie van de koppeling tussen ORM en logisch geformuleerde dependency-structuren een actueel onderwerp binnen de afdelingen Informatie en KennisSystemen (IRIS) en Informatics for Technical Applications (ITA).

Omdat ORM en systeemontwerp middels logisch geformuleerde dependency-structuren de belangrijkste componenten zijn van de brug tussen de zachte en de harde kant is het essentieel om stil te staan bij de koppeling hiertussen. De koppeling tussen ORM en logisch geformuleerde dependency-structuren wordt in de volgende tabel uiteengezet:

Object Role Modeling	Systeemontwerp middels logisch geformuleerde dependency-structuren
	**Systeemonderdeel** DEF Object := ... Van de objecten wordt een definitie gemaakt, van assumptions niet.
	**Logische symbolen** Verplichte rol-connector wordt in de logisch geformuleerde dependency-structuren aangeduid middels het logische symbool $\wedge$
	**Logische symbolen** Niet-verplichte rol-connector wordt in de logisch geformuleerde dependency-structuren aangeduid middels het logische symbool $\vee$
	**Voedingsstof systeemonderdeel** DEF Object := Attribuut $\wedge$ Attribuut $\vee$ ... etcetera Attributen zijn assumptions in logisch geformuleerde dependency-structuren; hoe deze geproduceerd worden, wordt niet weergegeven in logisch geformuleerde dependency-structuren omdat een attribuut het laagste niveauin ORM is.

Object Role Modeling	Systeemontwerp middels logisch geformuleerde dependency-structuren
	**Logische symbolen** Relaties tussen objecten en tussen objecten en attributen worden weergegeven middels logische symbolen: ¬ Niet ∧ En ∨ Of → Als ..., dan ... ↔ Dan en slechts dan als De keuze voor het symbool hangt sterk af van de betekenis van de rol. Daarnaast verlies je informatie omdat een rol een betekenis geeft aan een relatie in tegenstelling tot de summiere betekenis van een logisch symbool.
	**Kwantoren** N : N word weergegeven middels een ∀-kwantor. De 1 : 1, N : 1 en 1 : N worden weergegeven middels de ∃-kwantor
	**Systeemonderdelen** De enrollment tussen de objecten A en B zorgt ervoor dat de eigenschappen van A en B gecombineerd worden in de relatie met object C. DEF Object C := Object A ∧ Object B → Object C

D5 Conclusie

Middels deze methodologie is gebleken dat een zacht onderwerp als de analyse van enterprise architectuur formeel vastgelegd kan worden. Door gebruik te maken van ORM en systeemontwerp middels logisch geformuleerde dependency-structuren lijkt een brug geslagen te zijn tussen de “zachte” en de “harde” kant van het onderzoeksgebied.

De overgang van de definities vanuit de natuurlijke taal naar ORM-modellen is te verwezenlijken door gebruik te maken van het ORM-stappenplan. Om van de ORM-modellen naar logisch geformuleerde dependency-structuren te komen kan gebruik gemaakt worden van het dependency-stappenplan.

Door deze brug lijken ook de informatica- en informatiekundewereld nader tot elkaar gekomen te zijn. Deze brug is nu nog een touwbrug maar biedt mogelijkheden om van deze brug in de toekomst een stevigere brug te maken.

Dit zou kunnen geschieden door in de toekomst:

- De koppeling tussen de definities vanuit natuurlijke taal en ORM grondiger te onderzoeken en de ORM-modellen nader te verfijnen
- De koppeling tussen ORM en logisch geformuleerde dependency-structuren grondiger te onderzoeken en te bewijzen
- De logisch geformuleerde dependency-structuren wiskundiger te maken en te bewijzen

Er kan tevens geconcludeerd worden dat naarmate je verder de brug op gaat er informatieverlies optreedt. Door het exacter maken van de zachte kant gooi je informatie overboord. Het is daarom van essentieel belang om gedurende het wiskundiger maken van het onderzoeksgebied de reeds gevonden informatie hierbij te blijven betrekken.

Een andere conclusie is dat wanneer je de logisch geformuleerde dependency-structuren formuleert je de reeds gemaakte ORM-modellen controleert. Hierdoor kan het voorkomen dat de ORM-modellen bijgesteld moeten worden. Andersom geldt dit uiteraard ook. De kwaliteit van de opgeleverde modellen wordt hierdoor beter.

Literatuurlijst

[BEMELMANS]
Bemelmans T.M.A. (1991), *Bestuurlijke informatievoorziening en automatisering,* Kluwer Bedrijfswetenschappen, Deventer

[BENTHEM]
Benthem J.F.A.K., Ditmarsch H.P., Ketting J., Meyer-Viol W.P.M., (1991), *Logica voor informatici*, Addison-Wesley Nederland

[BERG]
Berg M., (2003), *Slopen onder architectuur,* Atos Origin, Sogeti, Centrum voor Wiskunde en Informatica. Lac 2003

[BOONSTRA]
Boonstra A., (2002), *ICT,mensen en organisaties, een management benadering,* Pearson Education

[CAST]
Casti J.L., (1979), *Connectivity, Complexity and Catastrophre,* John Wiley, New York.

[CORN]
Cornacchio J.V., (1977), *System complexity –A bibliography.,* Int. J. General Systems

[DALE]
Van Dale, *Woordenboek Nederlandse taal*
http://www.vandale.nl/opzoeken/woordenboek/

[EDMONDS]
Edmonds B., *What is complexity?*, Manchester Metropolitan University

[IEEE]
"IEEE Recommended practice for architectural description of software intensive systems (IEEE-std-1471-2000)." IEEE Standards Collection, Software Engineering, IEEE, New York

[ISES]
Pruijt L. J., Lommers J.A.P., *Productgerichte architectuur* ISES International
http://www.ises-international.com

[GIA]
Genootschap voor informatiearchitecten (1998, 6 november), *Definitie van informatiearchitectuur*
http://www.gia.nl

[GIANOTTEN]
Gianotten M.*, IT Turnaround, De visies en ervaringen van Nederlandse topmanagers*

[HALPIN]
Halpin T. (2001), *Information Modeling and Relational Databases; from conceptual analysis tot logical design*, Morgan Kaufmann Publishers

[HEYLIGHEN]
Heylighen F., (2002), *Complexiteit en Evolutie*, Vrije Universiteit Brussel

[HOV]
van der Hoven D., *Architectura universalis,* HIT
http://www.serc.nl/lac/docs/Papers/

[LEEUW]
Leeuw, A.C.J. de, (1982), *Organisaties: Management, analyse, ontwerp en verandering; een systeemvisie*, van Gorcum, Assen

[META1]
Meta Group, Appel W. (2003), *when failure is not an option, enterprise architecture – the next tsunami,* presentatie landelijk architectuur congres,

[MINTZ]
Mintzberg H., (2003), *Organisatiestructuren,* Academic Service

[PANFOX]
Van der Sanden W., Sturm B.(1997) *Informatiearchitectuur,de infrastructurele benadering*, Panfox

[PASCOE]
Pascoe-Samson E.I., (1993) *Organisatie,Besturing en Informatie,* Kluwer Bedrijfswetenschappen, Deventer

[PROPER]
Proper H.A., (2003) *Informatiekunde, exacte vaagheid*, inaugurele rede, Katholieke Universiteit Nijmegen

[RIJS]
Rijsenbrij D., Schekkermans J., Hendrickx H, (2002), *Architectuur, besturingsinstrument voor adaptieve organisaties*, Lemma

[SNIJDERS]
Snijders J.H., de Groot C.T., de Serière J.H.W.M. (1997) *Informatiekunde 1,* Stenfert Kroese, Houten

[SOG]
Sogeti Nederland B.V., *Architectuur nieuwe stijl: DYA*
http://www.sogeti.nl

[SYS]
Bakker H., van den Berg M., van Deursen A., *Het systematisch op orde houden van de ICT-assetportfolio,* Atos Origin, Sogeti, Centrum voor Wiskunde en Informatica.
http://www.serc.nl/lac/2003/Papers/

[VAN DER POLS]
Van der Pols. R. (2003), *Nieuwe Informatievoorziening,* Academic Service, Schoonhoven

[WUPPER]
Wupper H. Mader A., (1999) *System design as a creative mathematical activity*,
Katholieke Universiteit Nijmegen

Afkortingenlijst

CGEY: Cap Gemini Ernst & Young;
CIO: Chief Information Officer;
EA: Enterprise Architectuur;
EAI: Entrerprise Application Integration;
EAR: Enterprise Architectuur Raamwerk;
GIA: Genootschap van Informatie Architecten;
IEEE: Institute of Electrical and Electronics Engineers;
IRIS: Informatie en Kennissystemen;
NIII: Nijmeegs Instituut voor Informatica en Informatiekunde;
ORM: Object Role Modeling;
PC: Personal Computer;

Lijst van figuren