

Van DEMO naar workflow management

E. Oren

Delft, 15 april 2003



Van DEMO naar workflow management

Afstudeerverslag Technische Informatica

E. Oren

Delft, 15 april 2003

Technische Universiteit Delft
Faculteit der Informatietechnologie en Systemen
Afstudeerrichting Ontwerpen van Informatiesystemen



Samenvatting

DEMO is een methodiek om organisaties te modelleren. Workflow management systemen zijn softwaresystemen die de aansturing van personen in een organisatie ondersteunen. Deze systemen werken op basis van een bepaalde beschrijving van de organisatie en de bedrijfsprocessen daarin. Onderzocht is hoe een workflow management systeem is ingericht dat werkt op basis van een DEMO-beschrijving van organisaties.

Workflow management (*wfm*) systemen zijn softwaresystemen die de aansturing en uitvoering van het bedrijfsproces ondersteunen. Volgens de Workflow Coalition moet een *wfm* systeem ondersteuning bieden voor het modelleren van de workflow, het uitvoeren van de workflow, en het samenwerken met gebruikers, externe applicaties en externe workflow engines. *Wfm* systemen werken op basis van een definitie van het bedrijfsproces, en helpen bij de uitvoering en aansturing daarvan door voor elke instantie van het bedrijfsproces taken toe te wijzen aan mensen, machines of informatiesystemen die deze taken kunnen uitvoeren.

De CAP-theorie (die de basis vormt van DEMO) beschouwt een organisatie als een discreet dynamisch systeem, bestaande uit actoren met een afgebakende verantwoordelijkheid die transacties met elkaar aangaan om bepaalde feiten tot stand te laten komen. Actoren zijn de enige actieve componenten van dit systeem, zij zijn dus de enige componenten die toestandsovergangen doen plaatsvinden. De actorcyclus toont hoe en waarom deze toestandsovergangen tot stand komen.

Door de actorcyclus te combineren met andere concepten uit de CAP-theorie is een (generiek) bedrijfssysteem gemodelleerd dat de innerlijke werking van elke willekeurige actor toont. Dit bedrijfssysteem beschrijft tegelijkertijd op abstracte wijze de werking van elke willekeurige organisatie, aangezien de actoren de enige actieve componenten daarvan zijn.

De informatiebehoefte van dit bedrijfssysteem is opgesteld en voor een deelverzameling daarvan is een informatiesysteem ontworpen dat die behoefte vervult. Dit informatiesysteem is getoetst aan de functionele criteria die gelden voor *wfm* systemen. Met uitzondering van samenwerking met externe applicaties en externe workflow engines voldoet het ontworpen systeem aan de gestelde eisen. Deze uitzonderingen zijn naar verwachting zonder problemen toe te voegen aan het ontwerp.

Het ontworpen informatiesysteem is geïmplementeerd in een prototype. Het technisch ontwerp en de prototype implementatie zijn wellicht niet optimaal voor de ontwikkeling tot commercieel product, maar tonen wel aan dat het functionele ontwerp valt te realiseren. Ten aanzien van de verdere ontwikkeling zijn aanbevelingen gedaan: ten eerste hoe het prototype technisch afgemaakt moet worden, alvorens het als commercieel product de markt op zou kunnen, en ten tweede hoe het functionele ontwerp uitgebreid zou kunnen worden om werkelijk als *wfm* systeem te kunnen worden gekwalificeerd.

Voorwoord

Voor u ligt mijn afstudeerverslag, het resultaat van acht maanden werk aan mijn afstudeeropdracht. Met dit werk hoop ik de laatste fase van mijn studie af te ronden, en blijk te geven van de kennis die ik daarbij heb opgedaan. Ongeveer een jaar geleden kwam ik voor het eerst bij Jan Dietz met de vraag of hij wellicht een afstudeeropdracht voor mij had. Wetenschappelijk opgevoed en gevormd met de DEMO-methodiek had ik het idee dat DEMO vooral werd gebruikt voor het ontwerp van bedrijfsprocessen; ik wilde wel eens proberen of het niet ook op andere terreinen ingezet zou kunnen worden. Juist op het gebied van workflow management zou DEMO de critici kunnen bewijzen dat het niet slechts een abstract denkkader is over organisaties, maar dat het ook een heel concrete invulling kan geven aan informatiesystemen. In de tussentijd heb ik gelukkig gezien dat mijn eerste ideeën over de reeds uitgewerkte toepassingen van DEMO wat beperkt waren; het was zeer motiverend om te zien hoe mensen proberen deze methodiek in allerlei richtingen te gebruiken, bijvoorbeeld voor het opstellen van functionele eisen, het in detail ontwerpen van informatiesystemen of het samenbrengen en laten samenwerken van verschillende organisaties.

Ik ben ModulOr Technology dankbaar voor de tijd en ruimte die zij mij boden. Ik hoop van harte dat de concrete opbrengsten van dit project voor hen nuttig zullen zijn, en ik wens ze alle succes toe met het uitwerken van het prototype tot een commercieel product. Het is mijn stellige overtuiging dat workflow management op basis van DEMO een toekomst zou kunnen hebben, en ik hoop voor ModulOr Technology dat zij een rol kunnen spelen bij het ontwikkelen van producten in deze richting.

Bij ModulOr Technology studeerde rond dezelfde tijd ook mijn mede-student Martijn Stellinga af. Het is zeer waarschijnlijk dat onze veelvuldige en eindeloze discussies over DEMO, informatiesystemen en het ontwikkelen van software de productiviteit van andere werknemers negatief heeft beïnvloed; desondanks ben ik Martijn zeer dankbaar voor zijn niet aflatende bereidheid om zich over mijn modellen te buigen en daarmee de kwaliteit van mijn werk (en het plezier erin) duidelijk te verhogen.

Afstuderen is ook in bepaalde opzichten een lijdensweg, heen en weer geslingerd tussen de druk om nog meer dingen nog verder uit te werken, en de veelvuldig opkomende behoefte te zeuren, te stoppen en het op te geven. Eliz, bedankt, voor het aanhoren van al mijn verhalen over rode, groene en blauwe kabouters, voor het opmonteren elke keer, en voor de bijtijds noodzakelijke schop onder mijn kont.

Inhoudsopgave

1	Inleiding	15
1.1	Opdracht	15
1.2	Onderzoeksvraag	16
1.3	Onderzoeksopzet	16
1.4	Wetenschappelijke relevantie	16
2	Workflow management systemen	19
2.1	Organisatiemodellering	19
2.2	Inleiding en voorbeeld	20
2.3	Definities en uitleg	23
2.4	Functionele eisen aan <i>wfm</i> systemen	24
3	Workflow in DEMO concepten	29
3.1	Inleiding	29
3.2	Een organisatie model in DEMO termen	29
3.2.1	Eenvoudig model	31
3.2.2	Uitbreiding van concepten	33
3.3	Compleet model van bedrijfssysteem	35
3.3.1	Coördinatie-diagram	36
3.3.2	Feitendiagram	38
4	Functioneel ontwerp van een <i>wfm</i> systeem	45
4.1	Inleiding	45
4.2	Ondersteuning aan het bedrijfssysteem	45
4.2.1	Het informatiesysteem speelt actorrollen	45
4.2.2	Het informatiesysteem levert informatie uit productiebanken	48
4.2.3	Het informatiesysteem bewaart informatie	50
4.3	Functioneel ontwerp	51
4.4	Toetsing aan functionele eisen <i>wfm</i> systemen	53
4.4.1	Vergelijking met WfMC eisen	55
4.4.2	Vergelijking met andere <i>wfm</i> systemen	58
5	Prototype: technisch ontwerp en implementatie	65
5.1	Inleiding	65
5.2	Softwarestructuur	67
5.2.1	Decompositie in subsystemen	68
5.2.2	Verantwoordelijkheden binnen subsystemen	73
5.2.3	Ontwerpbeslissingen binnen subsystemen	78

5.3	Informatiestructuur	86
5.3.1	ORM-model	86
5.3.2	ERD-model	89
5.4	Implementatie: beschrijving en screenshots	93
5.5	Aanbevelingen bij verdere ontwikkeling	100
6	Conclusie	105
A	Diagrammen	vii
B	Stored Procedures	xi
C	Introductie tot DEMO	xvii
C.1	Language/ Action perspectief	xvii
C.2	CAP-theorie	xviii
C.2.1	Beginselen	xviii
C.2.2	Hiërarchische bouwstenen: atomen, moleculen en materie	xx
C.3	DEMO modelleertechniek	xxiii
C.3.1	Coördinatiediagram	xxiv
C.3.2	Processtapdiagram	xxv
C.3.3	Feitenmodel	xxvii

Lijst van figuren

2.1	WfMC Referentiemodel: componenten en interfaces [28]	25
2.2	WfMC Referentiemodel: componenten van generieke product structuur [28]	26
3.1	Flowchart van innerlijke werking actor	31
3.2	Coördinatiediagram bedrijfssysteem	32
3.3	Het uitvoeren van een coördinatieve actie [16]	34
3.4	Coördinatiediagram van het bedrijfssysteem	36
3.5	Coördinatiediagram van het bedrijfssysteem, inclusief interstrictie	37
3.6	Productiefeiten uit de externe feitenbanken	40
3.7	Voorbeeld ORM-model	41
3.8	Voorbeeld FCO-IM-model	41
3.9	Productiefeiten van de systeemtransacties	43
4.1	Functioneel ontwerp informatiesysteem	54
5.1	Functioneel ontwerp informatiesysteem, kopie van figuur 4.1	69
5.2	Decompositie in subsystemen	72
5.3	De subsystemen BUSINESSMODELMANAGER en PERSONNELMANAGER	75
5.4	De subsystemen PROCESMANAGER en TIMEMANAGER	76
5.5	Het subsysteem DATABASEMANAGER	77
5.6	Voorbeeld coördinatiemodel bibliotheek	80
5.7	Voorbeeld feitenmodel bibliotheek	81
5.8	Gebruikersinterfaces	82
5.9	Klassen in de BUSINESSMODELMANAGER	83
5.10	Klassen in de PROCESMANAGER en PERSONNELMANAGER	84
5.11	Klassen voor het databeheer	87
5.12	Informatiestructuur typeniveau	90
5.13	Informatiestructuur instantieniveau	91
5.14	Databaseschema ActieManager	92
5.15	Hoofdscherf ActieManager	93
5.16	DatabaseBeheer ActieManager	94
5.17	Personeelsmanagement ActieManager	95
5.18	Tijdsmanagement ActieManager	96
5.19	Externe feiten scherm ActieManager	97
5.20	Agendascherf ActieManager	97
5.21	Geselecteerd agendum ActieManager	98
5.22	Agendascherf ActieManager (initiëren van een interactie)	99
5.23	Propositie scherm ActieManager	99

A.1	Feitenmodel van het bedrijfssysteem, bij hoofdstuk 3	viii
A.2	Feitenmodel van het informatiesysteem, bij hoofdstuk 5	ix
A.3	Database schema (in ERD) van het informatiesysteem, bij hoofdstuk 5	x
C.1	Coördinatie-diagram bibliotheek, uit [14]	xxv
C.2	Processtapdiagram bibliotheek, uit [14]	xxvi
C.3	Feitendiagram bibliotheek	xxviii

Lijst van tabellen

4.1	Coördinatieve feiten vormen agendum voor performer of addressee . .	49
4.2	Informatiebehoefte van actorrollen uit het bedrijfssysteem, vloeit voort uit de interstrictie	51
4.3	Behoeft tot het bewaren van informatie, vloeit voort uit informatiebehoefte uit tabel 4.2	52
4.4	Totale informatiebehoefte	52
4.5	Verband functionaliteit informatiesysteem en informatiebehoefte actorrollen	53
5.1	Type gebruikers en de actorrollen daarin	70

Hoofdstuk 1

Inleiding

1.1 Opdracht

ModulOr Technology is een softwarebedrijf in Rotterdam dat producten levert die zich richten op kwaliteitsverbetering in organisaties en bedrijfsprocessen. Het bedrijf levert hiervoor onder andere de ModulOr Designer, een softwarepakket om DEMO-modellen te maken. Klanten kunnen met de ModulOr Designer hun organisatie volgens de DEMO-methodiek modelleren; de resulterende modellen geven de klanten inzicht in de eigen organisatie en de bedrijfsprocessen daarin. Deze modellen kunnen de basis vormen voor bijvoorbeeld ISO-certificering of interne kwaliteitsbewaking.

ModulOr Technology levert ook een ander softwarepakket om de kwaliteit in organisaties te verbeteren, namelijk de ActieManager. Dit pakket is bedoeld om goede communicatie in bedrijfsprocessen te bevorderen en de afzonderlijke processtappen te bewaken in die bedrijfsprocessen. De ActieManager is geïmplementeerd voor een klein aantal bedrijfsprocessen, door deze te analyseren en hun structuur in de software op te nemen. De ActieManager kan, bij organisaties waarin deze bedrijfsprocessen voorkomen, de processtappen bewaken.

Op dit moment is de ActieManager niet generiek, slechts het kleine aantal bedrijfsprocessen dat geïmplementeerd is wordt ondersteund. Voor elk ander bedrijfsproces dat organisaties willen beheren, moet het systeem worden uitgebreid. Dit betekent dus dat voor elke klant waarvan de organisatie niet precies de reeds geïmplementeerde bedrijfsprocessen heeft, een nieuw ontwikkeltraject moet worden gestart. Daarnaast is de functionaliteit van de ActieManager niet uitgebreid genoeg om als volwaardig workflow management systeem te kunnen worden ingezet.

De ActieManager en de Designer zouden elkaar goed kunnen aanvullen, indien de ActieManager generiek zou zijn en zou werken op basis van een DEMO-model van de organisatie. De ActieManager zou dan precies de bedrijfsprocessen ondersteunen die in de Designer zijn gemodelleerd. Het voordeel daarvan is ten eerste dat de ActieManager niet meer voor elk bedrijfsproces aangepast hoeft te worden en ten tweede dat de ActieManager dan direct aansluit op het organisatiemodel dat in de Designer is opgesteld.

Het verzoek van ModulOr Technology is om een nieuwe versie van de ActieManager te ontwerpen. De nieuwe ActieManager moet een functionaliteit aanbieden die meer in de buurt komt van een workflow management systeem en werken op basis van de DEMO-beschrijving van organisaties. De wens is om een generieke ActieManager

te ontwerpen die, uit het DEMO-model van elke willekeurige organisatie, een workflow management systeem zou kunnen genereren. Dat systeem zou vervolgens de workflow in de bedrijfsprocessen van die organisatie moeten ondersteunen.

1.2 Onderzoeksvraag

De opdracht is dus om een ontwerp te maken van een generieke ActieManager die bedrijfsprocessen ondersteunt op basis van een DEMO-model. Daarom is de onderzoeksvraag van dit werk als volgt geformuleerd: hoe is een workflow management systeem ingericht dat werkt op basis van een DEMO-beschrijving van een organisatie?

Hiervoor moeten we eerst onderzoeken of het mogelijk is om de functionaliteits-eisen die aan een workflow management (*wfm*) systeem worden gesteld af te leiden uit het DEMO-model van die organisatie. Vervolgens is de vraag hoe dat functionele ontwerp is te realiseren in een informatiesysteem.

1.3 Onderzoeksopzet

Om deze onderzoeksvraag te beantwoorden splitsen we haar in drie deelvragen, die achtereenvolgens beantwoord worden. Ten eerste is het de vraag wat precies een *wfm* systeem is, of waaraan een systeem moet voldoen om de workflow binnen een organisatie op een goede manier te ondersteunen. Deze vraag wordt in hoofdstuk 2 beantwoord, hetgeen leidt tot functionele criteria aan *wfm* systemen.

De tweede deelvraag is hoe uit een DEMO-beschrijving van een organisatie een functioneel ontwerp is op te stellen voor een *wfm* systeem. Dit functionele ontwerp moet aan alle (opgestelde) criteria voor *wfm* systemen voldoen, waarmee het zich kwalificeert als een *wfm* systeem. Dit wordt besproken in de hoofdstukken 3 en 4. Het antwoord op deze vraag laat zien dat een DEMO-model in ieder geval genoeg informatie bezit om er een functioneel ontwerp voor een *wfm* systeem uit op te stellen.

De laatste deelvraag is of het mogelijk is om dat functionele ontwerp ook daadwerkelijk te realiseren in een werkend systeem. Met het opstellen van het functioneel ontwerp op zich is namelijk nog niet aangetoond dat het maken van een *wfm* systeem mogelijk is, maar alleen dat de specificaties daarvoor zijn af te leiden. Aan specificaties die niet te realiseren zijn hebben we echter niets. Daarom toont hoofdstuk 5 dat realisatie mogelijk is, door het technisch ontwerp en implementatie van een prototype te beschrijven. Het geeft daarmee antwoord op de laatste deelvraag.

Het onderzoek beantwoordt dus, door middel van drie deelvragen, de vraag 'hoe richten we een *wfm* systeem in dat werkt op basis van een DEMO-model'. Het technisch ontwerp en de implementatie van het prototype zijn daarbij de concrete opbrengsten voor de opdrachtgever.

1.4 Wetenschappelijke relevantie

Van der Aalst stelt in [2] dat, aangezien workflow management systemen volledig afhankelijk zijn van expliciete modellen van de organisatie en de bedrijfsprocessen daarin, het belangrijk is om de juiste technieken te gebruiken bij het opstellen van deze modellen. Vervolgens stelt hij dat er reeds vele technieken gebruikt worden om bedrijfsprocessen te modelleren, maar dat deze geen van allen een geformaliseerde semantiek hebben. Hierdoor zijn ze ambigu en daarmee onbruikbaar als basis voor een workflow

management systeem. Van der Aalst stelt daarom voor om Petri-netten te gebruiken als modelleertechniek voor bedrijfsprocessen omdat deze een volledig geformaliseerde semantiek bevatten, goed analyseerbaar zijn, en goed bruikbaar zijn voor het modelleren van bedrijfsprocessen.

Het uitgangspunt van dit onderzoek is dat Van der Aalst in zijn constatering volledig gelijk heeft: het is nodig om een goede modelleertechniek te gebruiken bij het opstellen van de modellen die de basis vormen van workflow management systemen. We verwachten echter dat voor het opstellen van deze modellen DEMO een betere methodiek is dan Petri-netten. In tegenstelling tot Petri-netten modelleert DEMO niet slechts de bedrijfsprocessen, maar de essentie van de volledige organisatie. Daarnaast maakt DEMO voor het modelleren van de bedrijfsprocessen juist gebruik van Petri-netten, en is op dat aspect dus even geformaliseerd als Petri-netten.

Maar de hypothese dat DEMO beter gebruikt zou kunnen worden om de organisatie te modelleren dan Petri-netten, kan pas worden onderzocht als is aangetoond dat DEMO modellen überhaupt als invoer kunnen dienen voor een workflow management systeem. Op die vraag is daarom dit onderzoek gericht, of er dus een workflow management systeem kan worden gemaakt dat werkt op basis van DEMO modellen.

Van der Aalst stelt vast dat een goede modelleertechniek nodig is om modellen te maken voor workflow management systemen. Dit onderzoek wil laten zien dat DEMO hiervoor *bruikbaar* is. Ander onderzoek zal moeten uitwijzen of DEMO op dit gebied ook *beter* is dan andere methodieken.

Hoofdstuk 2

Workflow management systemen

2.1 Organisatiemodellering

De DEMO methodiek biedt een manier om organisaties te modelleren. Het model van een organisatie brengt de organisatie terug tot haar essentie. Het model toont de productie die de organisatie aan de buitenwereld levert, de verantwoordelijkheden van allen die bij die productie betrokken zijn, en de coördinatie die plaatsvindt om die productie mogelijk te maken. De basis van de DEMO methodiek is het idee dat een organisatie bestaat uit individuen die samenwerken om een bepaald doel te bereiken.

Het modelleren van organisaties kan om een aantal verschillende redenen plaatsvinden, die met elkaar gemeen hebben dat op de een of andere manier geprobeerd wordt de werking van de organisatie te verbeteren. Mogelijke toepassingen van organisatiemodellering, uit [21], zijn bijvoorbeeld kwaliteitsbewaking (ISO-certificering) of herontwerp van bedrijfsprocessen (BPR). In beide voorbeelden probeert men de werking van de organisatie te verbeteren: in het geval van ISO-certificering door de huidige processen in kaart te brengen en te toetsen aan gestandaardiseerde criteria; in het geval van BPR door de huidige processen te veranderen zodat knelpunten worden opgelost.

Deze voorbeelden laten zien hoe de werking van organisaties verbeterd kan worden door organisaties te modelleren en vervolgens te sleutelen aan het ontwerp van de bedrijfsprocessen. Bij het verbeteren van een organisatie kan men ook op een indirecte manier gebruik maken van organisatie-modellering, namelijk als niet gekeken wordt naar het ontwerp van de bedrijfsprocessen, maar naar de uitvoering van (het reeds opgestelde) ontwerp.

Deze benadering probeert dus niet het ontwerp van de bedrijfsprocessen te optimaliseren maar een zo correct en efficiënt mogelijke uitvoering van het afgesproken bedrijfsproces te realiseren. Het optimaliseren van de uitvoering van bedrijfsprocessen vindt natuurlijk in alle organisaties op de een of andere manier plaats, bijvoorbeeld bij een manager die probeert op de werkvloer zijn werknemers zo aan te sturen dat het bedrijfsproces goed uitgevoerd wordt.

Het geheel van mensen, procedures, gegevens en afleidingsregels dat betrokken is bij de uitvoering van bedrijfsprocessen is op te vatten als een informatiesysteem, waarin bijvoorbeeld managers aan werknemers bepaalde informatie leveren over de door hen te verrichten werkzaamheden. Een deel van dat informatiesysteem, namelijk het

beheer over bepaalde gegevens en de afleidingsregels ten aanzien van die gegevens, kan uitgevoerd worden door een bepaald type softwaresystemen, zogeheten workflow management (*wfm*) systemen. Deze systemen delen taken in het bedrijfsproces toe aan personen en zorgen ervoor dat die personen de informatie krijgen die ze voor het uitvoeren van de toegewezen taak nodig hebben. Niet alle bedrijfsprocessen zijn te ondersteunen door *wfm* systemen: zowel het werk als de gegevens moeten goed gestructureerd zijn. Per geval moet daarom bekeken worden of invoering van een *wfm* systeem mogelijk en wenselijk is.

Om *wfm* systemen te kunnen gebruiken is het noodzakelijk om op de één of andere manier de organisatie (en haar bedrijfsprocessen) te modelleren; immers, om de uitvoering van een bedrijfsproces te ondersteunen moet het systeem dat bedrijfsproces wel kennen. In dit onderzoek kijken we of DEMO hiervoor bruikbaar is, dus of DEMO modellen van organisaties gebruikt kunnen worden om bruikbare *wfm* systemen te maken. Om deze vraag te beantwoorden definiëren we eerst een *wfm* systeem en daarna wat de functionele eisen zijn aan een dergelijk systeem.

2.2 Inleiding en voorbeeld

Een *wfm* systeem is een softwaresysteem dat de uitvoering en aansturing van bedrijfsprocessen ondersteunt. Voordat we daarvan een meer formele definitie proberen te geven, introduceren we een voorbeeld van een *wfm* systeem:

Er is een verzekeringsmaatschappij die autoverzekeringen aanbiedt. Bij de verzekeringsmaatschappij kunnen twee soorten verzoeken telefonisch binnenkomen: verzoeken voor het afsluiten van een nieuwe (of gewijzigde) polis en verzoeken voor het uitkeren van schadevergoedingen. In beide gevallen noteert de telefoniste het verzoek op een daartoe bestemd formulier.

Bij verzoeken voor een nieuwe polis stuurt ze het formulier naar een acceptant. Deze ontvangt het formulier en beslist over acceptatie op basis van allerlei gegevens over de aanvrager en de auto. Bij een positieve beslissing stuurt de acceptant het formulier naar de actuaaris die de premiehoogte vaststelt. Vervolgens stuurt de administratie een brief naar de aanvrager met de mededeling dat de polis wel of niet is geaccepteerd; in het geval dat de polis is geaccepteerd sluiten ze er een acceptgiro bij in met het verzoek om de eerste premiebetaling tot stand te brengen.

Bij verzoeken voor schadevergoeding stuurt de telefoniste het formulier naar een schade-expert. Deze neemt beslissing over uitkering van de schade op basis van de polisvoorwaarden, de premiebetalingen van de verzekerde en eventuele extra informatie, zoals het politierapport. Ook nu wordt de klant door de administratie op de hoogte gebracht van de beslissing. Daarnaast stuurt de administratie eventueel een betalingsopdracht naar de bank om de berekende schadevergoeding uit te keren.

In dit eenvoudige voorbeeld zijn twee bedrijfsprocessen beschreven. Een *wfm* systeem zou het uitvoeren en aansturen hiervan moeten ondersteunen. Om de werking van een dergelijk *wfm* systeem te beschrijven introduceren we eerst een aantal elementen uit de literatuur (bijvoorbeeld [20]) die in dit voorbeeld aan de orde komen:

Casus Een *wfm* systeem draait altijd om het afhandelen van zogeheten casussen. Een casus kan zijn, zoals in dit voorbeeld, het afsluiten van een nieuwe polis of het uitkeren van een schadevergoeding, maar ook een patiënt die medisch getest moet worden of een klacht die afgehandeld moet worden. Een casus is een ‘zaak’ waar het bedrijfsproces over gaat en die tijdens het bedrijfsproces behandeld wordt.

Taak In het voorbeeld worden een aantal werkzaamheden genoemd die tijdens het bedrijfsproces moeten worden uitgevoerd. Een taak is gedefinieerd als een (voor de opdrachtgever) atomaire (ondeelbare) hoeveelheid werk. De taken die in dit voorbeeld uitgevoerd moeten worden zijn onder andere het nemen van een beslissing over de polisacceptatie, het schrijven en/of sturen van een brief over de acceptatie, het berekenen van de premie of het verzenden van de acceptgiro. Een taak kan zowel materieel (het verzenden van een brief) als immaterieel zijn (het berekenen van de premie). Een taak kan daarnaast zowel door een persoon worden uitgevoerd (het nemen van een beslissing over polisacceptatie) als door een informatiesysteem (het berekenen van de premie) of een machine (het sorteren van binnengekomen post). Een persoon, machine of informatiesysteem die een taak kan uitvoeren noemen we een *resource*. Het atomair zijn van een taak houdt in dat deze, voor de opdrachtgever, door één resource wordt uitgevoerd.

Bedrijfsproces Een bedrijfsproces bestaat uit een aantal taken en de volgorde waarin deze moeten worden uitgevoerd. In het voorbeeld zien we bijvoorbeeld dat na het ontvangen van een verzoek om schadevergoeding eerst wordt gekeken of de schade wordt gedekt door de polisvoorwaarden. Als dit niet het geval is dan wordt het verzoek afgewezen. Als dit wel het geval is wordt er gekeken of de verzekerde wel zijn premiebetalingen op orde heeft.

De beschrijving van het bedrijfsproces bestaat dus uit de opsomming van de taken die moeten worden uitgevoerd en de volgorde waarin dat moet gebeuren. Er zijn een aantal verbanden mogelijk tussen de verschillende taken, zoals sequentieel, causaal, conditioneel en selectief (zie voor een overzicht [5]). Een causaal verband (oorzaak en gevolg) betekent dat het uitvoeren van de ene taak leidt tot het uitvoeren van een andere taak. Een sequentieel verband betekent dat twee taken elkaar opvolgen. De twee afzonderlijke taken “noteren van een verzoek om schadevergoeding” en “oordelen over dat verzoek” hebben bijvoorbeeld een sequentieel verband: na het noteren van het verzoek wordt het verzoek beoordeeld. Een conditioneel verband betekent dat de uitvoering van een taak afhankelijk is van meer dan één andere taak. Al deze andere taken zijn dan condities voor het uitvoeren van die ene taak. Een selectief verband betekent dat er een keuzemogelijkheid zit in het procesverloop. In het voorbeeld is de beslissing over acceptatie van de premie of vergoeding van de schade een keuzemogelijkheid. Afhankelijk van de uitkomst van die taak wordt óf de ene taak uitgevoerd (berekening van de premie) óf de andere taak (verzending van een afwijfsbrief).

In [20] worden de verschillende verbanden tussen de taken niet als onderdeel van het bedrijfsproces gezien, maar als apart concept *routing*. Het is volgens ons niet zinvol om dit concept apart te onderscheiden, en bij een *bedrijfsproces* dus niet te spreken over de verbanden tussen de taken. Daarom zullen we *routing* niet als afzonderlijk concept behandelen maar als onderdeel van de definitie van het bedrijfsproces. Het bedrijfsproces bestaat dus uit de definitie van de afzonderlijke taken en de verbanden daartussen.

wfm systeem De taak van een *wfm* systeem is, zoals gezegd, het ondersteunen van de aansturing en uitvoering van het bedrijfsproces, het doel is over het algemeen niet het automatiseren van de taken in het bedrijfsproces [24]. Een *wfm* systeem is een softwaresysteem dat een deel van de informatielogistiek die een rol speelt bij het uitvoeren van het bedrijfsproces op zich neemt. In sectie 2.3 zullen we een meer formele definitie van *wfm* systemen uitwerken. Ter inleiding schetsen we kort waaruit in dit voorbeeld een *wfm* systeem zou bestaan:

Als er een aanvraag binnenkomt voor een nieuwe polis, dan noteert de telefoniste deze nog steeds op een formulier, en geeft vervolgens in het *wfm* systeem aan dat er een nieuwe polisaanvraag is binnengekomen. Ze refereert daarbij aan de formuliercode van het zojuist door haar ingevulde formulier. Het *wfm* systeem bepaalt vervolgens dat deze aanvraag doorgestuurd moet worden naar de acceptant, en kiest uit alle acceptanten die bij de verzekeringsmaatschappij in dienst zijn een acceptant, bijvoorbeeld diegene die op dat moment het minste werk te doen heeft. Die acceptant ziet vervolgens in zijn elektronische werkbakje dat hij een nieuwe polisaanvraag moet behandelen, met de bijbehorende formuliercode. Nadat de acceptant een beslissing heeft genomen vult hij dat in het *wfm* systeem in. Het systeem bepaalt vervolgens naar wie de aanvraag door moet worden gestuurd etc., etc., totdat het proces is afgerond.

Het is ook mogelijk, en over het algemeen efficiënter, om niet met papieren formulieren te werken, maar tegelijkertijd met het *wfm* systeem ook een documentbeheersysteem in te voeren waarin al deze gegevens worden genoteerd. Dan hoeven de werknemers niet meer formulieren aan elkaar door te geven.

Het *wfm* systeem stuurt dus de casussen door het bedrijfsproces naar de juiste resources. Telkens als een resource aangeeft een taak te hebben uitgevoerd bepaalt het *wfm* systeem wat er nu met de casus moet gebeuren, op basis van de gegeven specificatie van het bedrijfsproces. Het systeem zal dus, als de receptioniste aangeeft dat ze gereed is met het noteren van een acceptatieverzoek, bepalen dat er vervolgens een acceptatiebeslissing moet worden genomen, aangezien tussen deze taken een causaal verband is aangegeven. Het systeem doet niets anders dan de gegeven procesdefinitie volgen. Als het nemen van een acceptatiebeslissing ook conditioneel afhankelijk zou zijn van een andere taak, bijvoorbeeld van het bepalen of de verzekeringsmaatschappij wel mogelijkheden heeft om een nieuwe polis te accepteren, dan zou het systeem wachten op gereedkoming daarvan. De acceptant zou pas als ook die taak zou zijn afgerond de opdracht krijgen om een beslissing te nemen.

Het ondersteunen van de bedrijfsprocessen bestaat kortweg uit drie onderdelen: het bepalen van de taak die moet worden uitgevoerd, het bepalen van de resource die de taak zou moeten uitvoeren en het verschaffen van de benodigde informatie aan die resource. Het is belangrijk om te zien dat het *wfm* systeem slechts de proceslogica implementeert, dus voor elke individuele instantie van een casus bepaalt *welke* taken door welke resource uitgevoerd moeten worden. Het *wfm* systeem houdt zich echter niet bezig met de taaklogica, dus *hoe* de taken moeten worden uitgevoerd, en voert zelf ook geen taken uit [24].

2.3 Definities en uitleg

Er zijn in de loop van de jaren zeer veel implementaties van *wfm* systemen ontwikkeld, door allerlei verschillende partijen, waarbij de betrokkenen over het algemeen verschillende definities van workflow en *wfm* hanteerden. Hierdoor zijn deze pakketten niet vergelijkbaar en niet uitwisselbaar, hetgeen de invoering en acceptatie van *wfm* systemen schaadt. De Workflow Management Coalition [28] wil het gebruik en de ontwikkeling van *wfm* stimuleren door onder andere definities en specificaties te standaardiseren en een referentiemodel op te stellen. De Coalition wordt hierin gesteund door zowel de academische als de commerciële wereld. Een wijd geaccepteerde en gebruikte standaard heeft dit echter nog niet opgeleverd [1]; ondanks de grote aandacht voor *wfm* is er nog geen algemeen toepasbare theorie en conceptuele basis ontwikkeld [22]. Het werk van de Coalition zal daarom als uitgangspunt dienen voor ons onderzoek, en andere bronnen uit de literatuur zullen gebruikt worden om een breder idee te hebben van de verschillende opvattingen.

De Coalition stelt in [28] de volgende omschrijvingen vast:

workflow is de automatisering van een (gedeelte van een) bedrijfsproces waarbij documenten, informatie of taken volgens bepaalde procedures van de ene naar de andere deelnemer worden doorgegeven.

Een *wfm* systeem is een systeem dat de uitvoering van workflows definieert, creëert en beheert, door het gebruik van software die de procesdefinitie kan interpreteren, en die met gebruikers en externe applicaties kan samenwerken.

De interpretatie van deze definities, en het uitgangspunt bij het modelleren van bedrijfsprocessen, is afhankelijk van de invulling die men geeft aan de term “bedrijfsproces”. Volgens [18] zijn er twee basiscategorieën in het modelleren van bedrijfsprocessen: communicatie-gerichte modellering en activiteit-gerichte modellering. De communicatie-gerichte modellering van bedrijfsprocessen gaat uit van communicatiepatronen tussen zogenoemde “customers” en “performers”, die plaatsvinden omdat de customer behoefte heeft aan bepaalde goederen en/ of diensten die de performer kan leveren. In [23] worden deze conversaties tussen customer en performer, waarbij de laatste een actie uitvoert ter voldoening van de eerste, *workflow loops* genoemd. Elk bedrijfsproces bestaat uit een verzameling van zulke workflow loops. Communicatie-gerichte modellering van bedrijfsprocessen gaat er van uit dat alle acties het gevolg zijn van communicatie (en de afspraken die daaruit volgen), en ziet een bedrijfsproces dus als een verzameling van communicatiepatronen. Activiteit-gerichte modellering daarentegen, richt zich juist op het werk dat plaatsvindt in plaats van de afspraken die daar de oorzaak van zijn. Een bedrijfsproces wordt beschouwd als een verzameling taken die in een bepaalde volgorde moeten worden verricht.

In [1] wordt een activiteit-gerichte benadering gevolgd: workflows worden gedefinieerd als case-driven bedrijfsprocessen. Deze hebben drie dimensies: proces (de volgorde en definitie van taken die uitgevoerd moeten worden), resource (de mensen en machines die het werk kunnen uitvoeren, ingedeeld in groepen en rollen) en case (de specifieke instantie van een ‘zaak’ die behandeld moet worden). Een taak voor een specifieke case heet een *workitem*, en een *workitem* die is toegewezen aan een specifieke resource heet een *activity*. Dit is een meer werkbare definitie dan de Coalition hanteert, die veel van deze termen als synoniemen beschouwt.

In [23] wordt een communicatie-gerichte benadering gevolgd: workflow wordt gedefinieerd als de “interweaving of action workflow loops”. De taken die in een be-

drijfsproces moeten plaatsvinden zijn juist gedefinieerd door de requests die customers plaatsen in alle afzonderlijke workflow loops. [23] stelt dat traditionele workflow bestaat uit de volgorde van acties die op een object moeten worden uitgevoerd, waarbij coördinatie slechts één van de vele typen taken is die moeten worden uitgevoerd; volgens [23] zijn de taken juist gedefinieerd door de coördinatie.

Hierin verschillen deze twee benaderingen: waar een activiteit-gerichte benadering alle activiteiten gelijk behandelt, wordt in een communicatie-gerichte benadering de coördinatie als uitgangspunt genomen; alle taken worden bepaald door de coördinatie. De taken worden dus ook in communicatie-gerichte benaderingen beschouwd, maar niet als losstaande activiteiten maar als uitvloeisel van de coördinatie.

In [6] wordt een overzicht gegeven van *wfm* systemen waarbij een meer praktische, technisch gerichte, aanpak wordt gehanteerd. [6] stelt dat workflow software een infrastructuur biedt om bedrijfsprocessen te ontwerpen, uitvoeren en beheren; workflow software is een informatietechnologie die feedback verschaft en mogelijkheden biedt om bedrijfsprocessen te monitoren en te beheren. Taak automatisering is daarbij zeer goed mogelijk maar geen voorwaarde om voordeel te behalen uit het gebruik van *wfm* systemen.

[18] onderscheidt drie niveaus van processen: materiële (het assembleren van fysieke componenten en afleveren van fysieke goederen), informationele (geautomatiseerde taken die informatie creëren, verwerken, beheren en verschaffen) en bedrijfsprocessen (markt-gerichte beschrijvingen van de activiteiten van de organisatie). Bedrijfsprocessen worden daarbij geïmplementeerd door (een combinatie van) informationele of materiële processen en bestaan om een bedrijfscontract of klant-wens te vervullen. Een workflow wordt in [18] op twee niveaus gedefinieerd: het is de (conceptuele) beschrijving van taken uit het bedrijfsproces waarmee bedrijfsprocessen kunnen worden begrepen, beoordeeld en herontworpen. Het is echter ook de beschrijving van taken uit het informationeel proces zodanig, dat het de functionele eisen aan informatiesystemen en mensen specificiert. *Wfm* is vervolgens een technologie die het herontwerp van bedrijfs- en informationele processen ondersteunt. Volgens [18] is de taak van een *wfm* systeem dan ook ten eerste het bepalen en coördineren van de uitvoering van workflows, en ten tweede het zorgen voor een snel herontwerp en -invoering van de bedrijfs- en informatieprocessen.

Uit dit summier overzicht van enkele verschillende uitgangspunten ten aanzien van workflows en *wfm* systemen blijkt in ieder geval dat eenduidigheid en eenstemmigheid in het vakgebied ontbreken. Zoals gezegd zullen we juist daarom het werk van de Coalition als uitgangspunt nemen: niet omdat iedereen het erover eens is dat deze benadering de juiste is, maar omdat het een overeenstemming probeert te vinden tussen de verschillende uitgangspunten.

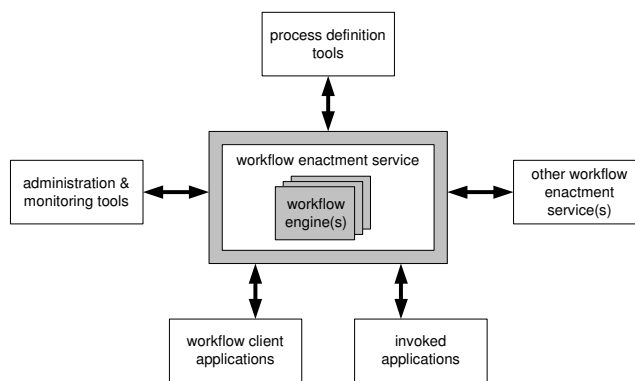
2.4 Functionele eisen aan *wfm* systemen

Om de functionaliteit van *wfm* systemen te kunnen vergelijken en de samenwerking ertussen te bevorderen, heeft de Coalition een referentiemodel opgesteld. Dit referentiemodel beschrijft ten eerste de noodzakelijke functionaliteit van een *wfm* systeem; het stelt daarbij dat het op drie gebieden ondersteuning moet bieden: bij het modelleren van de workflow, bij het uitvoeren van de workflow en bij het samenwerken met gebruikers en externe applicaties. Daarnaast beschrijft het referentiemodel de architectuur van een *wfm* systeem, door verschillende componenten te onderscheiden en interfaces daartussen te bepalen. Als makers van *wfm* systemen zich daaraan houden, dan kunnen hun

producten makkelijker met elkaar samenwerken.

In dit onderzoek wordt het referentiemodel van de Coalition niet gebruikt als uitgangspunt voor het ontwerp van een *wfm* systeem (het uitgangspunt voor het ontwerp is de DEMO-theorie) maar ter toetsing van het gemaakte ontwerp. Het referentiemodel biedt namelijk, in combinatie met definities uit [28], een beschrijving van functionele eisen aan *wfm* systemen. Aan die functionele eisen kunnen we ons ontwerp toetsen, en daarmee controleren in hoeverre we het ontworpen systeem als een *wfm* systeem kunnen kwalificeren.

Het referentiemodel is weergegeven twee figuren. Figuur 2.1 bestaat uit een overzicht van de componenten van een *wfm* systeem, en de interfaces die daartussen bestaan. Figuur 2.2 geeft in meer detail de innerlijke structuur van een *wfm* systeem aan.

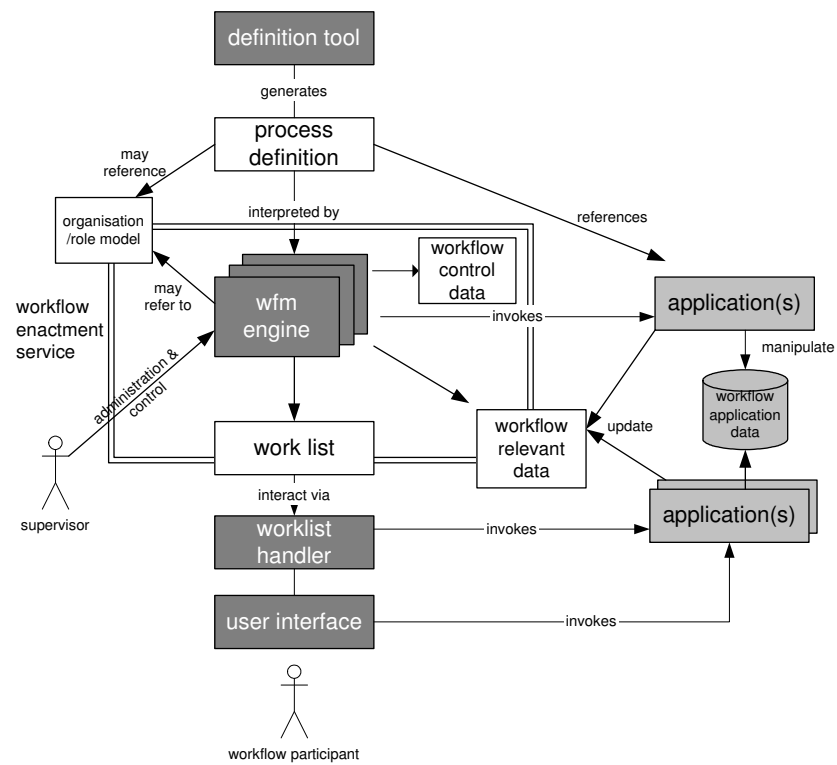


Figuur 2.1: WfMC Referentiemodel: componenten en interfaces [28]

In figuur 2.1 staat de workflow engine, de ‘motor’ van het *wfm* systeem, centraal. Deze wisselt gegevens uit met andere componenten, zoals de software voor het maken van procesdefinities, externe applicaties en andere workflow engines. In figuur 2.2 staat aangegeven welke informatie door de verschillende componenten gebruikt en gegenereerd wordt.

Zoals gezegd stelt de Coalition dat een *wfm* systeem op drie gebieden ondersteuning moet bieden aan de gebruikers: bij het modelleren van de workflow, het uitvoeren van de workflow en de samenwerking met gebruikers en externe applicaties. In figuur 2.2 zijn de drie componenten te vinden die deze ondersteuning bieden:

- *modelleergereedschap*: de component waarmee de definitie van het bedrijfsproces wordt opgesteld. Deze definitie moet uitvoerbaar zijn door een workflow engine. Het moet bestaan uit de activiteiten in het bedrijfsproces, de verbanden tussen de activiteiten en de gebruikers en applicaties die de activiteiten moeten uitvoeren. De definitie kan een verwijzing bevatten naar abstracte rollen uit een organisatie/rol model, in plaats van direct naar personen te verwijzen.
- *worklist handler*: beheert de interactie tussen de gebruiker en zijn worklist. Een gebruiker kan zowel een natuurlijk persoon zijn als een intelligent systeem. Een worklist bestaat uit de verzameling taken die aan een gebruiker zijn toegewezen. De worklist handler zorgt ervoor dat het *wfm* systeem taken kan toewijzen aan gebruikers en dat gebruikers statuswijzigingen in de taken aan het systeem door kunnen geven.



Figuur 2.2: WfMC Referentiemodel: componenten van generieke product structuur [28]

- *workflow engine*: biedt een runtime executie omgeving voor instanties van het bedrijfsproces. De workflow engine is het belangrijkste onderdeel van het *wfm* systeem. Dit component implementeert de proceslogica en wijst taken toe aan resources. De workflow engine interpreteert ten eerste de definitie van het bedrijfsproces. Ten tweede beheert het de verschillende instanties van het bedrijfsproces en navigeert daartussen. En ten derde wijst het voor elke instantie van het bedrijfsproces taken toe aan resources.

Om dit te kunnen beheert de workflow engine ten eerste de workflow control data, allerlei data over de status van verschillende instanties van het bedrijfsproces. Ten tweede beheert het de workflow relevant data, data die door externe applicaties wordt bewerkt. De workflow engine gebruikt deze data om te bepalen welke taken er uitgevoerd moeten worden. Ten derde meldt het gebruikers aan en af, zodat het weet welke gebruikers beschikbaar zijn.

Daarnaast biedt de workflow engine een aantal interfaces om samenwerking met andere applicaties mogelijk te maken waardoor bepaalde noodzakelijke functionaliteit aan kan worden geboden:

- het biedt een interface om externe applicaties aan te roepen en daar data mee uit te wisselen.
- het biedt een interface om met andere workflow engines samen te werken
- het biedt een interface om het bedrijfsproces te sturen en om toezicht te houden op het bedrijfsproces voor beheer, administratie en auditing.

Een *wfm* systeem moet dus op drie gebieden ondersteuning bieden. De Coalition heeft drie componenten gedefinieerd die deze ondersteuning moeten bieden. De beschreven functionaliteit van deze drie componenten vormt bij elkaar de functionaliteit van een *wfm* systeem.

Dit hoofdstuk heeft een overzicht gegeven van *wfm* systemen. Om aan te geven dat er in het vakgebied geen overeenstemming is over de precieze definitie en functionaliteit van een *wfm* systeem hebben we een kort overzicht gegeven van de verschillende standpunten op dit onderwerp. Om toch een toetsingskader te hebben voor ons eigen *wfm* systeem, hebben we het werk van de Coalition gebruikt. Een combinatie van het architecturele referentiemodel en de definitie van gebruikte begrippen heeft geleid tot een beschrijving van de algemene functionaliteit van een *wfm* systeem. Deze functionele eisen kunnen we later gebruiken om ons eigen *wfm* systeem te toetsen.

HOOFDSTUK 2. WORKFLOW MANAGEMENT SYSTEMEN

Hoofdstuk 3

Workflow in DEMO concepten

3.1 Inleiding

Elke organisatie is op drie abstractieniveaus [12] te beschouwen: essentieel, informatieel en documenteel. Op het essentiële niveau bestaat een organisatie uit subjecten met een bepaalde verantwoordelijkheid die taken uitvoeren en elkaars gedrag beïnvloeden; dit vormt een bedrijfssysteem. Op het informatiele niveau bestaat een organisatie uit een verzameling informatieverwerkers die informatie uitwisselen, bewaren, berekenen en afleiden; een informatiesysteem. Op het documentele niveau bestaat een organisatie uit operatoren die documenten creëren, transporteren, opslaan en vernietigen; een documenteel systeem. Tussen deze drie niveaus bestaat een hiërarchie: het documentele niveau is er ter ondersteuning van het informatiele niveau, en het informatiele niveau is er ter ondersteuning van het essentiële niveau.

Een informatiesysteem staat dus nooit op zichzelf, maar staat tussen een bedrijfssysteem (op het essentiële niveau) en documenteel systeem (op het documentele niveau) in. Het is juist het bedrijfssysteem dat een informatiebehoefte heeft, en het informatiesysteem dat die behoefte vervult, daarbij gebruik makend van een documenteel systeem. Voordat we een ontwerp opstellen van een informatiesysteem moet het bedrijfssysteem gemodelleerd worden, aangezien daar de informatiebehoefte en dus de functionele eisen vandaan komen.

Een *wfm* systeem is een informatiesysteem. Als we een ontwerp opstellen van een *wfm* systeem, moeten we dus eerst het bedrijfssysteem modelleren dat erdoor ondersteund moet worden. Een *wfm* systeem ondersteunt een werknemer bij het bepalen en uitvoeren van zijn taken. Het bedrijfssysteem dat door dit informatiesysteem ondersteund wordt, bestaat dus uit een werknemer die zijn taken bepaalt en uitvoert.

In dit hoofdstuk wordt het model van dit bedrijfssysteem geïntroduceerd. In het hierna volgende hoofdstuk zullen we een ontwerp opstellen van een informatiesysteem dat dit bedrijfssysteem ondersteunt. Dit informatiesysteem, de ActieManager, zullen we vervolgens toetsen aan de functionele eisen die in hoofdstuk 2 ten aanzien van *wfm* systemen zijn opgesteld.

3.2 Een organisatie model in DEMO termen

DEMO is een modelleertechniek voor organisaties, gebaseerd op de CAP-theorie. Voor lezers die daarmee niet of niet goed bekend zijn is als appendix C een inleiding in

DEMO en de CAP-theorie opgenomen; het wordt aangeraden deze inleiding eerst te lezen.

Volgens de CAP-theorie is een organisatie een sociaal systeem dat is ontworpen om producten aan de omgeving te leveren [10]. Deze producten kunnen zowel materieel als immaterieel zijn. De productie wordt tot stand gebracht door subjecten die bepaalde actorrollen vervullen. Deze actoren (een actor is een subject in een bepaalde actorrol) communiceren met elkaar om hun productie te coördineren. Deze coördinatie houdt in dat actorrollen afspraken met elkaar maken en zo elkaar aanzetten om gezamenlijk de productie aan de buitenwereld te leveren. De CAP-theorie ziet taal dus niet slechts als een middel om informatie over te brengen maar juist als een middel om een ander tot actie aan te zetten, doordat in de communicatie verplichtingen worden aangegaan (het Language/Action perspectief).

De CAP-theorie ziet een organisatie als een discreet dynamisch systeem. Een discreet dynamisch systeem bestaat uit actieve componenten, die acties kunnen uitvoeren. Als gevolg van die acties verandert de toestand van het systeem. Deze acties zijn instantaan en vinden op discrete tijdstippen plaats [10]. In de CAP-theorie zijn actoren de enige actieve componenten van een organisatie. Actoren voeren coördinatieve en productieve acties uit. De coördinatieve acties veranderen de toestand van de coördinatie wereld, de productieve acties veranderen de toestand van de productiewereld. Aangezien actoren de enige actieve componenten in het systeem zijn biedt de levenscyclus van een actor een totaalbeeld van hetgeen in een organisatie plaatsvindt. De levenscyclus van een actor beschrijft immers volledig de (zich continu herhalende) stappen die de actor uitvoert.

De levenscyclus van een actor bestaat uit drie stappen [12]: het selecteren van een agendum uit zijn agenda, het selecteren van een toepasselijke actieregel en het uitvoeren van die actieregel. De agenda van een actor is de verzameling van de coördinatieve feiten waarop de actor kan reageren. Op elk moment dat de actor 'niets te doen heeft' selecteert hij uit deze verzameling één agendum waarop hij wil gaan reageren. Dan volgt de tweede stap uit de actorcyclus: het selecteren van een actieregel. Een actieregel beschrijft voor een agendum (een bepaald coördinatief feit) hoe de actor zal reageren. Aangezien een actorrol bestaat uit de verzameling van zijn actieregels, moet hij bepalen welke van de beschikbare actieregels voor het eerder geselecteerde agendum relevant is. Vervolgens voert de actor de laatste stap uit van zijn levenscyclus, het uitvoeren van de geselecteerde actieregel. Dit bestaat uit het doen van één of meer coördinatieve en/ of productieve acties. De coördinatieve acties kunnen dan weer een agendum vormen voor een andere actor of eventueel voor deze zelfde actor. De levenscyclus begint dan weer opnieuw. In de actorcyclus zit het Language/Action paradigma besloten: geen enkele actor doet iets uit eigen beweging, maar altijd als reactie op een agendum. Zo'n agendum is altijd een coördinatief feit en dus het gevolg van een coördinatieve actie. Deze coördinatieve actie wordt gerealiseerd middels een communicatieve actie. Acties vinden dus slechts plaats als gevolg van, en als reactie op, coördinatie.

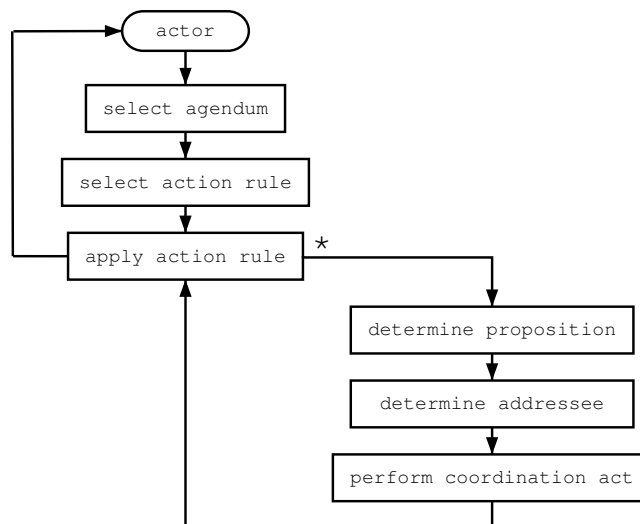
Voor het doen van een coördinatieve actie (onderdeel van de derde stap uit de levenscyclus) moet de actor een aantal dingen bepalen, zoals aan wie de coördinatieve actie gericht moet worden, wat de propositie is van de coördinatieve actie en welke intentie de actor ten aanzien van die propositie wil overbrengen. Een coördinatief feit bestaat namelijk uit de volgende onderdelen: een performer (degene die de actie uitvoert), een addressee (degene aan wie de actie is gericht), een propositie (een bepaalde toestand waarover de performer iets wil zeggen) en een intentie (hetgeen hij over de propositie zegt). Voor iedere coördinatieve actie die de actor, tijdens het uitvoeren van

een actieregel, wil doen moeten hij dus bepalen aan wie, waarover en wat hij iets wil zeggen.

3.2.1 Eenvoudig model

Deze beschrijving kunnen we omzetten in een DEMO-model, waarvan het coördinatie-diagram is gegeven in figuur 3.2. Dit model beschrijft de acties die een actor uitvoert. Aangezien actoren de enige actieve componenten zijn van een organisatie, beschrijft dit model daarmee ook alles wat er in een organisatie gebeurt. Het is een model van een actor die, in samenwerking met andere actoren, zijn verantwoordelijkheden neemt. Dit model gaat niet over één concrete actor of één concrete organisatie, maar beschrijft hoe alle actoren, in alle organisaties, hun werk doen.

Het systeem dat is gemodelleerd beschrijft dus de innerlijke werking van elke willekeurige actor. Een actor wordt vaak benaderd als een black-box, het enige dat we weten is dat hij steeds de actorcyclus doorloopt. Dit systeem modelleert de actor nu juist van binnen. Om verwarring te voorkomen is het model eerst, in figuur 3.1, in een flowchart beschreven. We zien een actor die tijdens zijn bestaan allerlei activiteiten uitvoert. Deze activiteiten komen overeen met zijn levenscyclus.

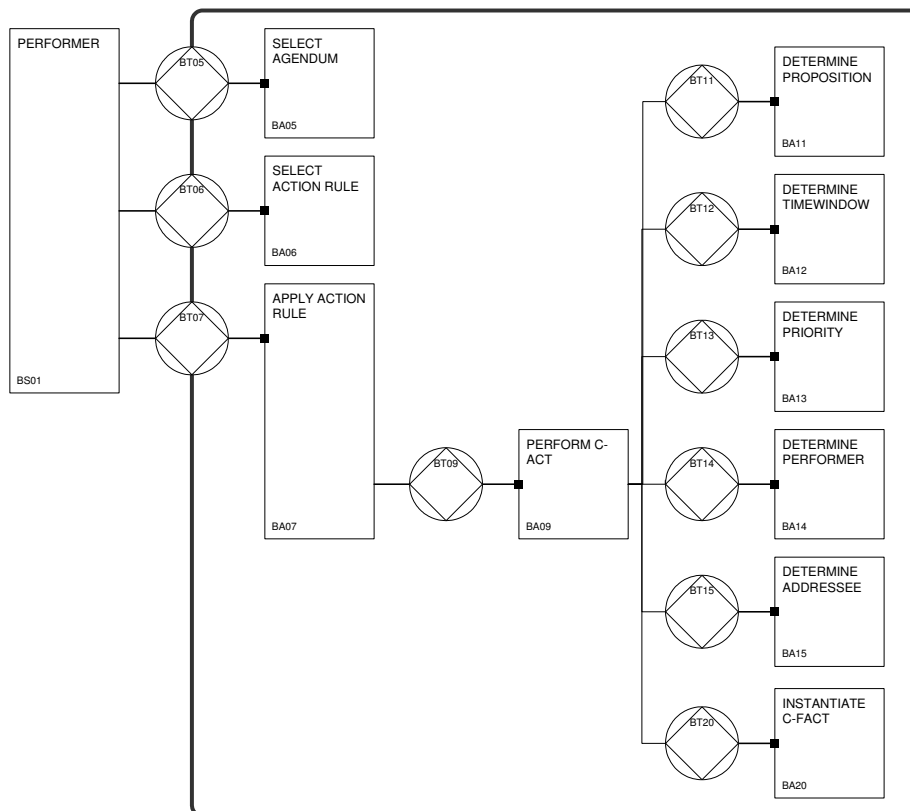


Figuur 3.1: Flowchart van innerlijke werking actor

In figuur 3.2 is ditzelfde systeem beschreven in een DEMO-model. Het voordeel van het modelleren van het bedrijfssysteem met behulp van DEMO is dat er een duidelijk verband kan worden gelegd met zowel de functionaliteit als de informatiehuishouding van het ondersteunende informatiesysteem, dat we later zullen ontwerpen.

De verschillende activiteiten binnen de actor, zoals het selecteren van een actieregel en het uitvoeren van een coördinatieve actie, zijn gemodelleerd als verschillende verantwoordelijkheden. Deze verschillende verantwoordelijkheden modelleren we, volgens de DEMO-methodiek, tot allerlei actorrollen, die met elkaar samenwerken. Het systeem is dus recursief: alle gemodelleerde actorrollen zitten 'in' elke actor; het systeem vormt in zijn geheel één actor.

In het coördinatie-diagram zien we dat een actor drie transacties kan initiëren die



Figuur 3.2: Coördinatiediagramm bedrijfssysteem

overeenkomen met zijn levenscyclus: het selecteren van een agendum, het selecteren van een actieregel en het toepassen van een actieregel. De transactie *apply action rule* bestaat uit het doen van één productieve en één of meer coördinatieve acties. Het uitvoeren van de productieve actie staat hierin niet opgenomen, omdat we slechts de coördinatiewereld modelleren. Het uitvoeren van een coördinatieve actie leidt tot een aantal transacties om te bepalen tegen wie, waarover, wat gezegd wordt, waarna vervolgens de actie daadwerkelijk wordt uitgevoerd, hetgeen resulteert in het instantiëren van een nieuw coördinatief feit.

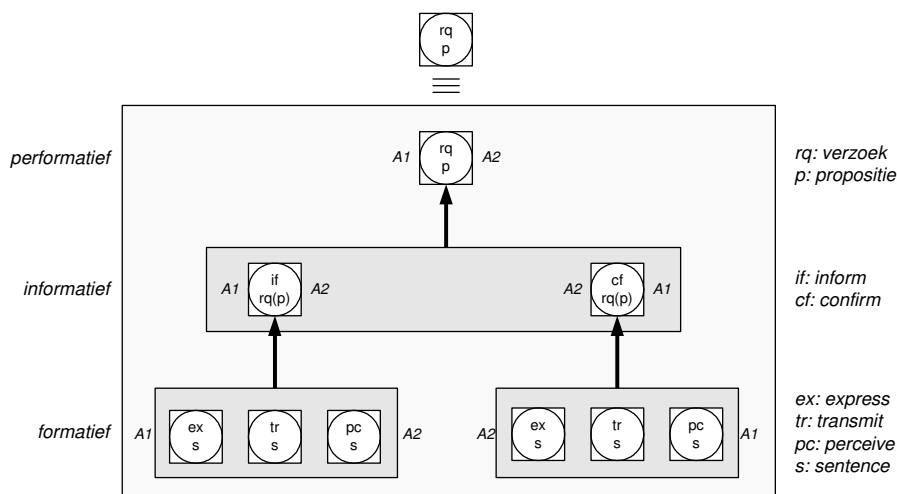
3.2.2 Uitbreiding van concepten

Het model in figuur 3.2 is een vereenvoudigde voorstelling van het bedrijfssysteem, en voor een bruikbaar model moeten drie concepten verder uitgewerkt worden. Ten eerste behelst het uitvoeren van een coördinatieve actie meer dan tot nu toe is gezegd. Ten tweede is het bepalen van de addressee van een coördinatieve actie geen triviale zaak, en is daarvoor informatie nodig die nu nog niet is beschreven. Ten derde is een belangrijk concept uit de DEMO-theorie niet beschreven, namelijk het *delegeren* van taken.

Deze drie aanvullingen worden nu beschreven, waarna we het complete model van het bedrijfssysteem, waarin deze elementen verwerkt zijn, zullen tonen.

Uitvoeren van een coördinatieve actie Volgens de CAP-theorie bestaat het uitvoeren van coördinatieve actie uit het uitvoeren van handelingen op drie semiotische abstractieniveaus (performatief, informatief en formatief), hetgeen in figuur 3.3 is te zien. In de figuur is getoond dat een coördinatieve actie, waarin A1 aan A2 verzoekt om een bepaalde propositie p , bestaat uit handelingen op drie niveaus. Tezamen vormen al deze handelingen de ene coördinatieve actie. Op het performatieve niveau bestaat de coördinatieve actie eruit dat A2 de sociale verantwoordelijkheid voelt om überhaupt op het verzoek van A1 te reageren (hetzij bevestigend, hetzij ontkennend). Op het informatieve niveau zijn er voor de coördinatieve actie twee handelingen nodig: A1 moet A2 zijn verzoek overbrengen en A2 moet vervolgens aan A1 doorgeven dit verzoek te hebben correct gekregen. Deze informatieve handelingen van informeren en bevestigen hebben een drager nodig; alle informatie heeft een drager nodig. Daarom moet voor elke informatieve handeling een zin, die de inform of confirm voorstelt, worden opgesteld en geuit. Deze zin moet dan naar de ander worden verzonden en daar vervolgens ontvangen worden.

Op elk van deze drie niveaus kan de communicatie mislopen. Als er op het performatieve niveau iets fout gaat (bijvoorbeeld een brief arriveert niet, een fax is onleesbaar of zit volledig vol met spelfouten) dan is de boodschap onleesbaar. Als dit wel goed gaat, maar er op het informatieve niveau iets fout gaat (als bijvoorbeeld de ontvanger van een Nederlandse brief geen Nederlands spreekt) dan is de boodschap (intellectueel) onbegrijpbaar. En als het ook op het informatieve niveau goed gaat, en de ontvanger dus precies begrijpt wat de boodschap is, kan het voorkomen dat hij zich niet aangesproken voelt om te reageren (bijvoorbeeld als een klant in een winkel per ongeluk een andere klant aanziet voor een medewerker), en er dus iets fout gaat op het performatieve niveau. Alleen als alle handelingen op al deze drie niveaus zijn geslaagd is de handeling 'A1 verzoekt A2 propositie p ' succesvol uitgevoerd. Dit verzoek kan vervolgens een agendum zijn voor een actorrol (bijvoorbeeld A2) die zijn reactie zal bepalen en uitvoeren.



Figuur 3.3: Het uitvoeren van een coördinatieve actie [16]

Bepalen van de adressee Een actorrol is een abstract begrip uit de CAP-theorie en stelt een rol in de organisatie voor. Een actorrol is gedefinieerd als een bepaalde verantwoordelijkheid, en de daarbij horende competentie en bevoegdheid, om één essentiële taak uit te voeren. Actorrollen worden gespeeld door personen, afdelingen of hele bedrijven. Een persoon in een organisatie vervult over het algemeen afwisselend meerdere actorrollen. Het toewijzen van een actorrol aan een persoon, waarbij die persoon de verantwoordelijkheid krijgt om die actorrol te vervullen heet *autorisatie* en vindt plaats bij indiensttreding, ontslag, promotie, en degradatie.

Een DEMO-model beschrijft actorrollen, de transacties die ze met elkaar aangaan en de communicatie waarmee dat gebeurt. Aangezien actorrollen in de werkelijkheid door mensen gespeeld worden, vindt de communicatie niet plaats tussen actorrollen, maar tussen mensen die deze actorrollen op dat moment vervullen. Het is dus niet zo dat een actorrol een coördinatieve actie uitvoert jegens een andere actorrol: een persoon die de ene actorrol speelt, voert een coördinatieve actie uit jegens een persoon die de andere actorrol speelt.

Actorrol A1 is producer van de transactie ‘goederen levering’. Actorrol S1 is customer van deze transactie. Adriaan kan actorrol S1 spelen, en zowel Bastiaan als Claus kunnen actorrol A1 spelen. Indien Adriaan deze transactie wil initiëren, en dus aan A1 de levering vragen van een bepaald goed, dan moet hij een persoon vinden in de organisatie die deze actorrol A1 kan spelen. Immers, actorrol A1 is een abstract begrip, daar kan Adriaan niet mee communiceren. Aangezien er in de organisatie twee personen de bevoegdheid hebben om A1 te spelen, Bastiaan en Claus, moet Adriaan kiezen aan wie hij zijn verzoek zal richten.

Maar als Adriaan helemaal niet weet wie er in dit bedrijf werken, of wie daarvan A1 mag spelen, dan kan hij een Adressee Bepaler, Ab, te hulp roepen. Adriaan vertelt Ab dat hij een verzoek aan A1 wil richten, en Ab geeft dan aan Adriaan door wie hij daarvoor moet hebben.

De *adressee bepaler* heeft informatie nodig om zijn taak te kunnen uitvoeren.

Ten eerste moet hij de verzameling kennen van personen die de gevraagde actorrol kunnen spelen. Ten tweede heeft hij, om deze personen optimaal te kunnen inzetten, ook informatie nodig over hun tijdsplanning en onverwachte afwezigheid (door ziekte etc.). Als de *addressee bepaler* weet wie al veel te doen heeft, en wie nog tijd over heeft, dan kan hij zorgen dat de performer zijn verzoek richt aan diegene die nog tijd over heeft. Om dit goed in te kunnen plannen, moet hij ook een schatting kennen van de tijd die nodig is om dit verzoek uit te voeren.

De *addressee bepaler* moet dus weten welke werknemers welke actorrollen kunnen spelen, hoeveel tijd werknemers beschikbaar hebben voor hun actorrollen (inclusief vakanties, ziekmeldingen etc.), welke taken ze op zich hebben genomen, en wat de geschatte tijdsduur is van alle taken. Zo kan hij de uitvoering van taken in de organisatie optimaal plannen en verdelen.

Delegatie Naast autorisatie is er nog een manier waarop subjecten (personen) volgens de CAP-theorie verantwoordelijkheid voor het spelen van een actorrol kunnen krijgen. De producer van een transactie kan namelijk een ander persoon vragen om een deel van zijn taak over te nemen:

Stel dat Bastiaan, in zijn rol van actor A1, aan Adriaan heeft beloofd om een propositie P tot stand te brengen. Als Bastiaan om welke reden dan ook, niet zelf de benodigde productieve actie kan of wil uitvoeren, kan hij een derde (Dirk) vragen dit te doen. Nadat Dirk dit heeft gedaan, kan Bastiaan aan Adriaan verklaren dat de propositie P tot stand is gebracht. Dat Dirk eventueel niet geautoriseerd is om actorrol A1 te spelen doet niet ter zake. Bastiaan is geautoriseerd om A1 te spelen, en hij draagt ook de verantwoordelijkheid.

Het is ook mogelijk dat Dirk bijvoorbeeld de verklaar-actie voor zijn rekening neemt. Dat Adriaan niets met Dirk, maar met Bastiaan, heeft afgesproken, doet ook wederom niet te zake. Dirk handelt namens Bastiaan, en Bastiaan blijft verantwoordelijk. Aangezien Dirk waarschijnlijk niet weet wat Bastiaan precies beloofd heeft, moeten deze twee wel met elkaar overleggen.

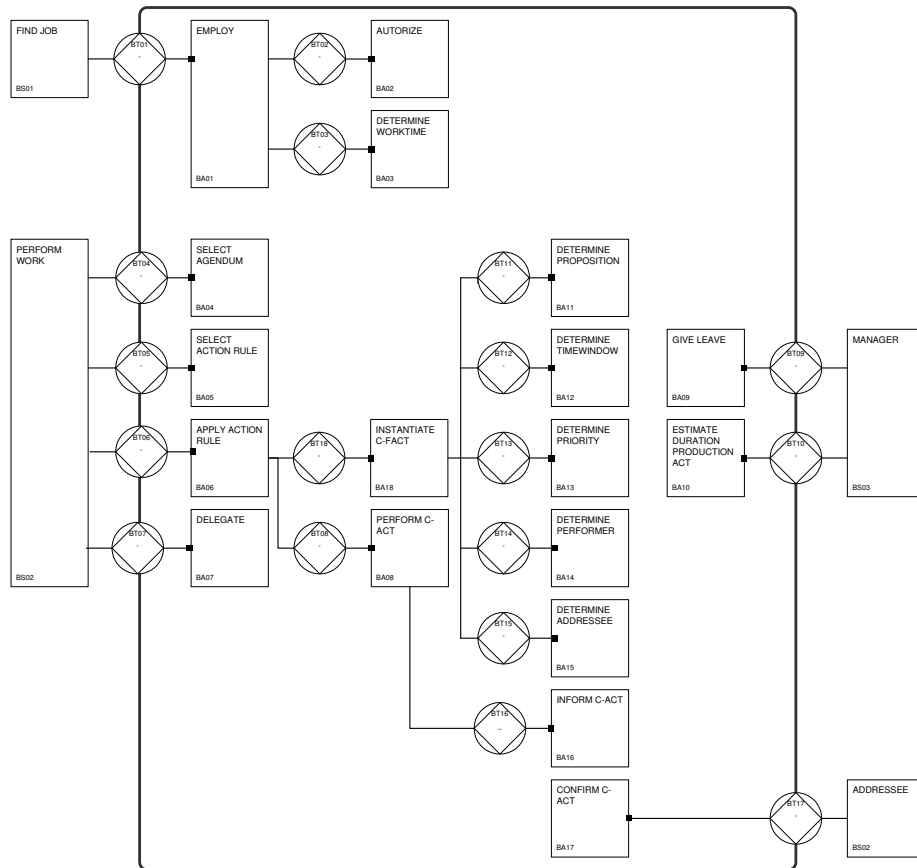
Een producer heeft altijd de mogelijkheid een deel van zijn taken te delegeren [16]. Hij blijft tegenover de customer wel de verantwoordelijkheid behouden. Aangezien de delegant (producer) aan de delegaat moet duidelijk maken wat hij moet doen, en volgens welke normen, vindt er tussen hen een tweede-orde coördinatie plaats. Waar eerste-orde coördinatie probeert de andere actorrol aan te zetten tot het uitvoeren van productieve acties, heeft tweede-orde coördinatie het harmoniseren van de normen van delegant en delegaat ten doel.

3.3 Compleet model van bedrijfssysteem

Uitbreiding van het model leidt tot het complete DEMO-model van het bedrijfssysteem. Het coördinatie-diagram daarvan is in figuur 3.4 weergegeven, en in figuur 3.5 is ook de interstrictie weergegeven. In het coördinatie-diagram is nu, naast een overzicht van de transacties die de actorrollen met elkaar aangaan, ook de interstrictie weergegeven die voor sommige actorrollen geldt. Deze interstrictie is aangegeven met een gestippelde lijn, van een actorrol naar een productiebank, en houdt in dat de actorrol wordt beïnvloed door de feiten in die productiebank. Concreet betekent dit dat de actorrol aan de

producer van die transactie informatie kan opvragen (in een informatieve transactie) over de productiefeiten die in die transactie tot stand zijn gebracht.

Eerst zal het coördinatiediagram worden besproken, aan de hand van de hiervoor behandelde concepten, daarna wordt het feitenmodel besproken. Bij de bespreking van het feitenmodel zullen de productiefeiten van alle transacties aan de orde komen.



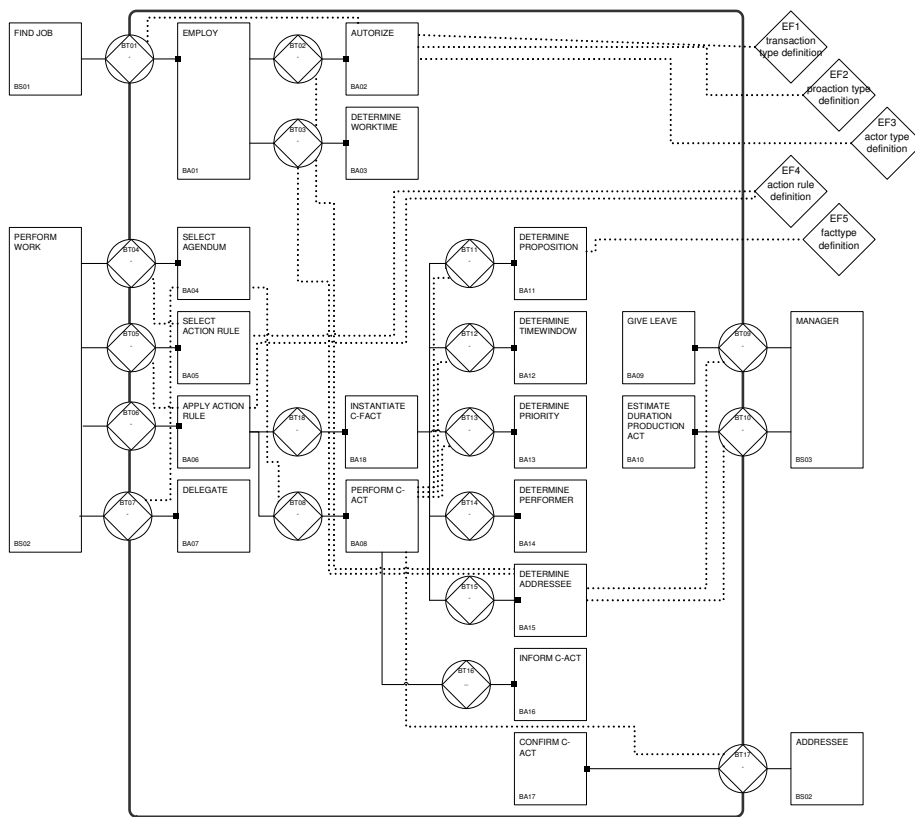
Figuur 3.4: Coördinatiediagram van het bedrijfssysteem

3.3.1 Coördinatiediagram

Ten opzichte van het eerste coördinatiediagram (figuur 3.2) zijn er een aantal dingen veranderd. Dit komt doordat de beschreven aanvullingen op een aantal concepten zijn verwerkt en de interstrictie is gemodelleerd.

Uitvoeren van een coördinatieve actie Ten eerste is vastgesteld dat een coördinatieve actie op het informatieve niveau bestaat uit twee handelingen: een inform en een confirm. Het bedrijfssysteem dat we modelleren bevindt zich op dit informatieve niveau van de coördinatieve actie. Een coördinatieve actie komt pas tot stand als de addressee heeft bevestigd dat hij deze heeft ontvangen en begrepen. Daarom is de transactie *store confirmation* gemodelleerd, die moet worden afgerond voordat de transactie *instantiate c-fact* kan worden afgerond.

3.3. COMPLEET MODEL VAN BEDRIJFSSYSTEEM



Figuur 3.5: Coördinatiediagram van het bedrijfssysteem, inclusief interstricte

Bepalen van de adressee Ten tweede hebben we gezien dat het bepalen van de juiste adressee van een coördinatieve actie niet triviaal is, maar één van de kerntaken van een *wfm* systeem. De actorrol *determine adressee* heeft informatie nodig over de autorisaties (wie mag welke actorrol spelen) en de tijdsplanning (wie heeft hoeveel tijd vrij) van alle werknemers. Dit is in het model weergegeven door de interstrictie die deze actorrol heeft met de productiebanken van een aantal transacties. Om te weten welke werknemers voor welke actorrollen geautoriseerd zijn inspecteert deze actorrol de productiebank van de transactie *authorize*. Om te weten hoeveel tijd de werknemers aan elke actorrol besteden inspecteert hij de productiebank van de transactie *determine worktime*. En om te weten welke werknemers verlof hebben (bijvoorbeeld vakantie of ziekte) inspecteert hij de productiebank van de transactie *give leave*. Tenslotte heeft deze actorrol ook informatie nodig over hoe lang in het algemeen bepaalde taken duren. Hiervoor bekijkt hij de productiebank van de transactie *estimate duration production act*.

Delegatie De mogelijkheid om als actor een taak te delegeren aan een andere persoon, is gemodelleerd in de transactie *delegate*. De bevoegdheid voor het delegeren van een agendum aan een persoon berust volledig bij de delegant. Het is niet noodzakelijk dat de delegaat geautoriseerd is voor de bewuste actorrol: de delegant neemt op eigen gronden de beslissing om een agendum te delegeren en blijft verantwoordelijk voor de correcte afhandeling ervan.

Interstrictie Tenslotte hebben ook een aantal andere actorrollen informatie nodig voor het uitvoeren van hun taak, ook zij hebben dus interstrictie met productiebanken van transacties. De actorrol *authorize* moet, om werknemers voor actorrollen te kunnen autoriseren, namelijk weten welke transacties, proacties en actorrollen er bestaan in de organisatie. Alleen dan kan hij een werknemer voor een actorrol autoriseren, als customer of producer van een transactie of een proactie. De drie feitenbanken waarmee hij interstricteert bevatten die informatie.

De actorrol *select agendum* kiest uit de verzameling agenda van de performer een agendum om af te handelen. Hij moet dus toegang hebben tot de verzameling agenda van de performer. Deze verzameling zit in de productiebank van de transactie *perform c-act*. Deze transactie leidt namelijk tot het ontstaan van allerlei coördinatieve feiten; de agenda van de performer is hiervan een deelverzameling, namelijk die coördinatieve feiten die voor deze performer een agendum vormen. Ook kijkt hij in de productiebank van de transactie *delegate*, om te kijken of er agenda aan de performer zijn gedelegeerd.

De actorrol *select action rule* bepaalt welke actieregel van toepassing is op het geselecteerde agendum. Daarvoor moet hij uiteraard de verzameling actieregels kennen van de performer, hetgeen in de externe feitenbank zit die deze actorrol raadpleegt. De actieregels zijn, net als de transacties en actorrollen, deel van het DEMO-model van de organisatie.

3.3.2 Feitendiagram

In het feitendiagram zijn, in ORM-notatie [19], de productiefeiten opgenomen van alle transacties in het systeem. Ook zijn in het feitenmodel de productiefeiten gemodelleerd die in de externe feitenbanken zitten. Ook deze zijn tot stand gekomen in transacties, maar deze transacties vallen buiten het systeem en zijn daarom onbekend. Van die

transacties zijn daarom alleen de productiebanken gemodelleerd: de externe feitenbanken.

Het diagram is in twee figuren gesplitst: in figuur 3.6 zijn de productiefeiten uit de externe feitenbanken getoond, in figuur 3.9 de productiefeiten van de systeemtransacties. Deze twee diagrammen sluiten op elkaar aan en vormen samen het feitenmodel. Het feitendiagram is in zijn geheel opgenomen in bijlage A.

Productiefeiten externe feitenbanken

In de externe feitenbanken staat het DEMO-model van de organisatie waarin de actor werkt, bijvoorbeeld een pizzeria of een verzekeringsmaatschappij. In deze feitenbanken staat beschreven welke transacties er in deze organisatie plaatsvinden, welke actorrollen daarbij betrokken zijn, en hoe de actieregels van die actorrollen gedefinieerd zijn. De structuur van deze feitenbanken is gelijk aan de structuur van elk DEMO-model. Deze structuur is het metamodel van DEMO, dat beschrijft welke concepten er voorkomen (bijvoorbeeld transacties en actorrollen) en wat de relatie tussen die concepten zijn. Het relevante, in de interstrictie voorkomende, gedeelte van dit metamodel is in figuur 3.6 beschreven.

Interactie Elke *interactie* heeft één producer en één of meer customers. Een interactie is óf een transactie, waarin een productief feit tot stand wordt gebracht, óf een proactie, waarin een actor wordt aangezet om een andere interactie aan te gaan. Een *transactie* bestaat uit een orderfase, een productiefase en een resultaatfase. De resultaatfase bestaat uit een productief actietype, dat resulteert in een productief feittype. De *proactie* en de order- en resultaatfasen van een transactie, zijn conversaties. Deze bestaan uit een aantal *coördinatieve actietypen*, die resulteren in coördinatieve feittypen.

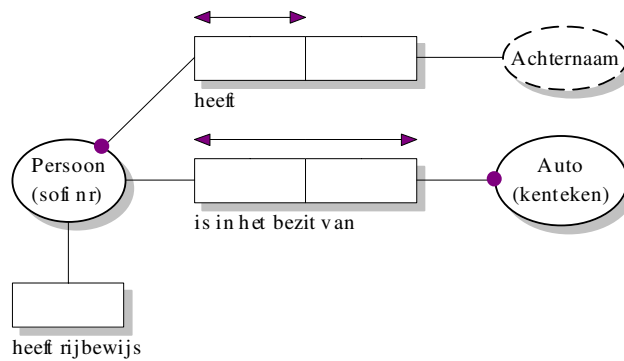
Coördinatief feittype Een *coördinatief feittype* bestaat uit een performer, addressee, intentie en propositie. De *performer* en *addressee* van een coördinatief feittype zijn bepaalde actorrollen. De *intentie* is er één uit de verzameling mogelijke intenties (zoals ‘verzoeken’, ‘beloven’ en ‘verklaren’). De *propositie* is óf een coördinatief feittype (bij een proactie), óf een productief feittype (bij een transactie).

Actorrol Een *actorrol* bestaat uit de verzameling van zijn actieregels. De inhoud van de actieregels is niet gemodelleerd, wel de entry- en exit-points ervan. De *entry-point* van een actieregel is het coördinatief feittype waarop de actieregel betrekking heeft, de *exit-points* van een actieregel zijn de coördinatieve feittypen die de actieregel tot gevolg heeft, als reactie op de entry-point.

Productief feittype Een productief feittype is gemodelleerd als combinatie van de manier waarop FCO-IM [8] en ORM feittypen modelleren. ORM onderscheidt drie concepten in een feitenmodel: feittypen, objecten en rollen. Een *feittype* is een predikaat met één of meer *rollen*, die door *objecten* gespeeld worden. In figuur 3.7 is ter illustratie een klein voorbeeld getoond, waarin drie feittypen gemodelleerd zijn. Het autobezit van een persoon, de achternaam van een persoon, en het rijbewijsbezit van een persoon. De eerste twee zijn binaire feittypen (met twee rollen), de laatste unair (bestaat uit één rol). Onder het feittype is geschreven hoe de verwoording in natuurlijke taal luidt, zoals “*Persoon ... is in het bezit van Auto ...*”. PERSOON en AUTO zijn entiteitstypen,

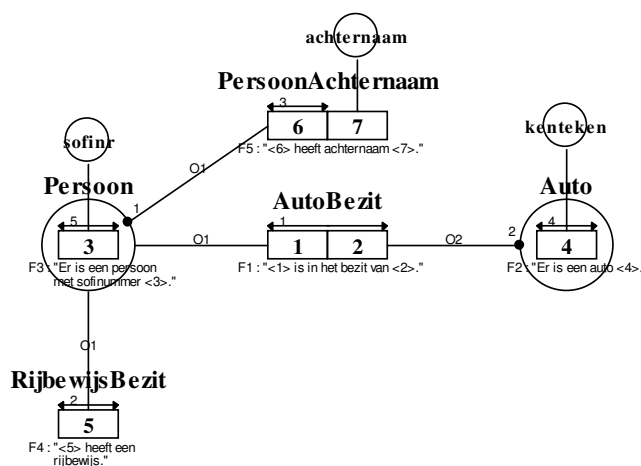
3.3. COMPLEET MODEL VAN BEDRIJFSSYSTEEM

concepten met een identiteit en eigenschappen; tussen haakjes staat genoteerd hoe naar deze entiteiten gerefereerd wordt. ACHTERNAAM is een waardetype, een verzameling lexicale begrippen zonder identiteit. Het is mogelijk om beperkingen (constraints) te definiëren ten aanzien van de populatie van een rol door een object. Een *uniciteitsregel* (streep over een rol) betekent dat een bepaalde populatie van die rol maar één keer mag voorkomen. Een persoon mag dus maar één naam hebben, maar meerdere personen kunnen dezelfde naam hebben; een persoon mag echter wel verschillende auto's bezitten. Een *totaliteitsregel* (zwart bolletje aan een entiteitstype) betekent dat de gehele populatie daarvan in het feittype moet voorkomen. Een persoon heeft dus altijd een achternaam, maar niet per se een rijbewijs of auto.



Figuur 3.7: Voorbeeld ORM-model

Ditzelfde voorbeeld is in figuur 3.8 met behulp van een FCO-IM-model beschreven. Het verschil met de ORM-notatie is dat in FCO-IM alle objecttypen een geobjectificeerd feittype zijn. Het objecttype **PERSON** bijvoorbeeld, is de objectificatie van alle feittypen "Er is een persoon met sofinummer ...". In FCO-IM is er geen onderscheid tussen feittypen en objecten: alles is een feittype. Rollen worden óf door een waardetype, óf door een geobjectificeerd feittype gespeeld.



Figuur 3.8: Voorbeeld FCO-IM-model

In het opgestelde DEMO metamodel (figuur 3.6) is een combinatie gemaakt van bovenstaande technieken. Een *productief feittyte* bestaat uit één of meer *rollen*. Deze rollen worden óf gespeeld door een productief feittyte, óf door een *waardetype*. Een rol staat op een bepaalde positie in een feittyte, en is eventueel onderhevig aan één of meer *beperkingsregels*. Het feittyte heeft een *reading* die aangeeft hoe de verwoording luidt in natuurlijke taal.

Productiefeiten systeemtransacties

In figuur 3.9 zijn de productiefeiten getoond van alle transacties in het coördinatiemodel. Dit maakt duidelijk waar de transacties over gaan, en wat ze tot gevolg hebben. De feittypen zijn overeenkomstig genummerd met de transactietypen waarvan ze het resultaat zijn: feittyte T01 is het resultaat van transactie T01.

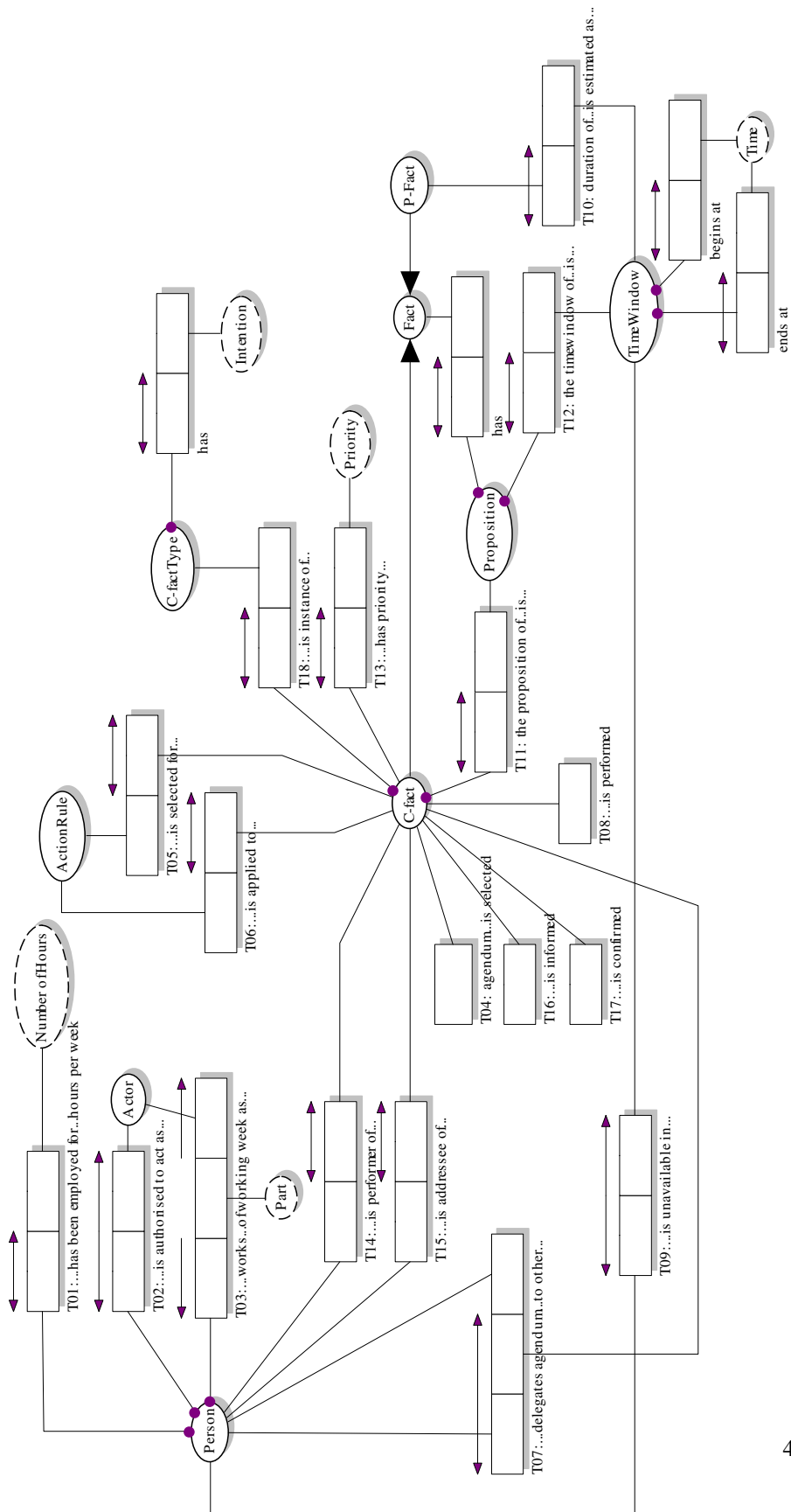
Autorisatie De transacties T01, T02 en T03 gaan over het in dienst treden van een nieuwe werknemer en de autorisatie van deze werknemer voor één of meerdere actorrollen. Van een werknemer is zowel bekend voor hoeveel uur hij in totaal bij de organisatie in dienst treedt, als hoeveel uur hij aan elke door hem vervulde actorrol besteedt. Een werknemer kan meerdere actorrollen vervullen, en een actorrol kan door meerdere personen vervuld worden.

Actorcyclus De transacties T04, T05 en T06 gaan over de handelingen van een actor tijdens zijn actorcyclus. Een coördinatief feit dat een agendum is voor deze actor wordt geselecteerd. Een actieregel die op dat feit van toepassing is wordt geselecteerd, en uitgevoerd. De uitvoering van een actieregel bestaat uit het doen van één of meer coördinatieve acties, en eventueel een productieve actie.

Coördinatieve acties Het uitvoeren van een coördinatieve actie staat gelijk aan het tot stand brengen van een coördinatief feit, hetgeen is gemodelleerd als het gevolg van transactie T08. Een coördinatief feit is een instantie, een specifiek specimen, van een coördinatief feittyte. Die instantie heeft een bepaalde inhoud, die eraan gegeven moet worden.

Het uitvoeren van een coördinatieve actie bestaat uit drie stappen: ten eerste het bepalen van de eigenschappen van het tot stand te brengen feit, hetgeen gebeurt in transacties T11 t/m T15; ten tweede het instantiëren van het coördinatieve feit met die inhoud, zodat erover gesproken kan worden, hetgeen gebeurt in transactie T18; en ten derde het informeren en laten bevestigen door de adressee van het bestaan van de overgebrachte informatie, hetgeen gebeurt in transacties T16 en T17. Met het instantiëren van het coördinatieve feit op zich is er nog geen coördinatief feit tot stand gebracht: de coördinatieve actie is pas geslaagd, en het coördinatieve feit is pas tot stand gebracht, als de adressee op informatieel niveau bevestigt dat hij de informatie heeft ontvangen.

Het uitvoeren van coördinatieve acties vindt plaats op instantie-niveau. Een coördinatief feit is een instantie van een coördinatief feittyte, de propositie ervan bestaat óf uit een productief feit (een instantie van een productief feittyte) óf uit een coördinatief feit. De performer en adressee van een C-feittyte zijn actorrollen, de performer en adressee van een C-feit personen. Immers, op het typeniveau kunnen we wel spreken over actorrollen die met elkaar communiceren, maar op instantieniveau zijn het altijd personen die, in een bepaalde actorrol, met elkaar communiceren. Een C-feit heeft



Figuur 3.9: Productieften van de systeemtransacties

HOOFDSTUK 3. WORKFLOW IN DEMO CONCEPTEN

geen intentie, die is namelijk niet eigen voor de specifieke instantie, maar voor alle instanties gelijk aan de intentie van het C-feittype.

Hoofdstuk 4

Functioneel ontwerp van een *wfm* systeem

4.1 Inleiding

In dit hoofdstuk wordt een ontwerp gepresenteerd van een informatiesysteem, de ActieManager, dat het bedrijfssysteem uit hoofdstuk 3 ondersteunt. Aangezien dit informatiesysteem ondersteuning moet bieden aan (een aantal) actorrollen uit het gemodelleerde bedrijfssysteem zal het ontwerp ervan afgeleid worden van de informatiebehoefte van deze actorrollen. Maar dit informatiesysteem is ook een *wfm* systeem, en zal daarom moeten voldoen aan de -in hoofdstuk 2 daaraan gestelde- functionele criteria.

Het informatiesysteem zal niet alle actorrollen uit het bedrijfssysteem ondersteunen, maar een deel daarvan. Eerst zullen we uitleggen en verantwoorden welke actorrollen, en dus welk deel van het bedrijfssysteem, door het informatiesysteem ondersteund zullen worden. Vervolgens zullen we uit het bedrijfssysteem de informatiebehoefte van de geselecteerde actorrollen afleiden. Deze informatiebehoefte leidt tot een functioneel ontwerp van het informatiesysteem: wat moet het systeem aan de gebruikers aanbieden, welke informatie hebben de gebruikers nodig.

We eindigen het hoofdstuk door de ontworpen functionaliteit van het informatiesysteem te vergelijken met de opgestelde functionele eisen ten aanzien van *wfm* systemen in het algemeen. Om het ontworpen informatiesysteem als een *wfm* systeem te kunnen kwalificeren, moet het namelijk voldoen aan die functionele eisen.

4.2 Ondersteuning aan het bedrijfssysteem

Het informatiesysteem biedt op twee manieren ondersteuning aan het bedrijfssysteem: ten eerste speelt het zelf een aantal actorrollen uit het bedrijfssysteem; ten tweede levert het informatie uit productiebanken aan actorrollen die deze informatie nodig hebben.

4.2.1 Het informatiesysteem speelt actorrollen

De transacties die in het bedrijfssysteem zijn gemodelleerd komen in elke organisatie voor. Om deze transacties uit te kunnen voeren moet elke actorrol uit het bedrijfssysteem door minstens één persoon binnen de organisatie worden vervuld. Soms worden

de actorrollen uit het bedrijfssysteem door alle werknemers vervuld, en soms door een daarvoor toegewezen werknemer. Zie bijvoorbeeld de actorrol *select agendum*:

In sommige organisaties hebben werknemers bij de uitvoering van de aan hun toegewezen verantwoordelijkheid de vrijheid om steeds zelf te beslissen welke van de op hen wachtende taken ze gaan uitvoeren. Er zijn ook organisaties waarin deze actorrol juist door een centraal persoon, zoals een afdelingsmanager, wordt gespeeld. In een dergelijke situatie kunnen medewerkers niet zelf kiezen welke taken ze gaan uitvoeren, maar maakt de afdelingsmanager voor hen die keuze.

Een aantal actorrollen uit het bedrijfssysteem kan door het informatiesysteem worden gespeeld. Dit kan slechts indien de actieregel (de werkprocedure) van die actorrol volledig bekend is. De actieregel is dan om te zetten in een algoritme, dat in een geautomatiseerd informatiesysteem kan worden uitgevoerd. Indien de actieregel niet bekend is, of technisch niet uitvoerbaar, dan kan de actorrol niet door een informatiesysteem gespeeld worden. Naast een technische reden, kan het ook, om bijvoorbeeld sociale of bedrijfskundige redenen, niet wenselijk zijn om een actorrol door een informatiesysteem te laten spelen.

Een opmerking over de taalkeuze moet hier gemaakt worden: hoewel we dat voor het leesgemak soms zeggen, kan een actorrol nooit echt door een informatiesysteem gespeeld worden. Een actorrol is immers een sociaal concept, een bepaalde verantwoordelijkheid om een taak uit te voeren. Een informatiesysteem is een apparaat dat weliswaar een actorrol kan naspelen, maar nooit de verantwoordelijkheid kan dragen van die actorrol. Een informatiesysteem is slechts een *agent*, die in opdracht van een persoon een actorrol imiteert. De persoon die een opdracht geeft aan het informatiesysteem behoudt altijd, samen met de personen die het informatiesysteem instrueren, de verantwoordelijkheid voor de handelingen van het systeem.

In dit geval zijn dat ten eerste de medewerkers van de organisatie waarin het informatiesysteem wordt gebruikt, die een deel van hun werkzaamheden aan het informatiesysteem delegeren. Ten tweede zijn het de medewerkers die hebben besloten dat het informatiesysteem ingevoerd moet worden, en daarmee de anderen medewerkers opdracht geven tot het overdragen van een deel van hun werkzaamheden aan het informatiesysteem. Ten derde zijn het de ontwerpers en makers van het informatiesysteem, die het informatiesysteem technisch instrueren, door bepaalde algoritmes te programmeren, en daarmee het gedrag van het informatiesysteem bepalen.

De medewerkers kunnen hun werkzaamheden bijvoorbeeld niet aan het informatiesysteem overdragen indien deze niet beschikbaar is; ze zullen dan toch zelf deze actorrollen moeten spelen en kunnen hoogstens de makers of de beleidsmakers op het probleem aanspreken. Ook als het informatiesysteem een vergissing maakt, bijvoorbeeld door een technische fout, een programmeerfout of een onvoorziene omstandigheid, en dus niet zo handelt als een medewerker zou doen, zijn de medewerkers zelf verantwoordelijk. Het kan nooit zo zijn dat de verantwoordelijkheid aan een informatiesysteem wordt overgedragen; uitspraken als “dat heeft het systeem besloten” of “dat is niet mijn verantwoordelijkheid, dat doet het systeem” zijn inherent onjuist.

Actorrollen die het informatiesysteem niet kan spelen De actorrollen BA01, BA02 en BA03 kennen aan personen in de organisatie een verantwoordelijkheid toe om een bepaalde actorrol in de organisatie te spelen. De actorrol BA07 delegeert een verantwoordelijkheid van één persoon naar een ander persoon, de actorrol BA09 verleent een

medewerker toestemming om een bepaalde tijdsperiode niet aanwezig te zijn en de actorrol BA10 maakt voor elke productieve actie een schatting van de tijd die de actie in beslag neemt.

Van deze zes actorrollen is het in eerste instantie niet gewenst dat ze door het informatiesysteem gespeeld zullen worden. De beslissing die daar genomen wordt valt ten eerste buiten de algemeen beschouwde functionaliteit van een *wfm* systeem, is ten tweede vrij lastig in een actieregel te beschrijven, en heeft ten derde veel informatie nodig die niet in een DEMO-model voorkomt (zoals de capaciteiten van een werknemer, of de financiële situatie van de organisatie).

Het is, om sociale redenen, niet wenselijk om de actorrol BA04 door het informatiesysteem te laten spelen. Mensen vinden het over het algemeen niet prettig als ze geen enkele invloed hebben op het werk dat ze moeten doen. Een informatiesysteem dat voor elke werknemer continu bepaalt wat deze moet doen wekt weerstand op. De actorrollen BA05 en BA06 zijn binnen dit onderzoek om technische redenen niet goed te automatiseren. Als randvoorwaarde is in dit onderzoek namelijk gesteld dat het DEMO-model van de organisatie dat aan het informatiesysteem beschikbaar is, slechts bestaat uit het coördinatiemodel en het feitenmodel. De actorrollen BA05 en BA06 hebben kennis nodig van de actieregels van alle actorrollen uit de organisatie. Aangezien deze kennis er niet is (we moeten het doen met het coördinatiemodel en het feitenmodel) kunnen deze actorrollen niet geautomatiseerd worden.

De gebruikers van het informatiesysteem zullen dus zelf de actorrollen A04, A05 en A06 moeten spelen. De informatiebehoefte die daaruit voortvloeit komt verderop aan de orde. Het betekent in ieder geval dat de gebruikers van het informatiesysteem een lijst te zien krijgen met hun agenda en daar zelf één agendum uit moeten selecteren. Vervolgens moeten ze zelf een actieregel selecteren die op dat agendum van toepassing is en deze toepassen.

Actorrollen die het informatiesysteem gedeeltelijk kan spelen De actorrollen BA11 t/m BA13 zijn gedeeltelijk door een informatiesysteem te vervullen, maar niet volledig. Het bepalen van de propositie, tijdsvenster en prioriteit van een coördinatieve actie valt bij een verzoekende coördinatieve actie niet door het informatiesysteem te doen. Dit is het gevolg van, hetgeen reeds is genoemd, het ontbreken van kennis over de actieregels. In de actieregels staat immers hoe door actorrollen op een agendum gereageerd wordt, en dus ook hoe de coördinatieve actie eruit ziet die als reactie wordt uitgevoerd.

De actorrollen BA11 t/m BA13 kunnen echter wel door een informatiesysteem vervuld worden als het gaat om niet-verzoekende coördinatieve acties, zoals een belofte, een verklaring of een acceptatie. In al die gevallen immers, is de propositie etc. van de coördinatieve actie in kwestie gelijk aan de propositie etc. van de verzoekende coördinatieve actie. Per definitie immers, bestaat een transactie uit een aantal coördinatieve acties die gaan over dezelfde propositie.

Actorrollen die het informatiesysteem wel kan spelen Er zijn een aantal actorrollen die het informatiesysteem volledig kan spelen, namelijk de actorrollen BA08, BA14 t/m BA16 en BA18. De actorrol BA14 is een vrij triviale actorrol, de performer van een coördinatieve actie is namelijk altijd de performer van de transactie T06. De actorrol BA16 is verantwoordelijk voor het informeren van de addressee over de coördinatieve actie, een taak die een informatiesysteem bij uitstek kan afhandelen. De actorrol BA18 legt het verband tussen een coördinatief feit en het coördinatieve feittype waarvan het de instantie is, dit kan een informatiesysteem doen aangezien het een formaliseerbare

afleidingsregel is. De actorrol BA08 voert een coördinatieve actie uit, en brengt daarmee een coördinatief feit tot stand; ook dit is een met een afleidingsregel te beschrijven. Al deze actorrollen zouden door het informatiesysteem gespeeld kunnen worden.

De actorrol BA15 is één van de belangrijkste actorrollen uit een *wfm* systeem, en kan zeker door een geautomatiseerd informatiesysteem vervuld worden. Deze actorrol bepaalt aan welke persoon een coördinatieve actie gericht gaat worden. Door te bepalen aan wie de verzoekende coördinatieve actie uit een transactie gericht is, bepaalt deze actorrol ook wie de productieve actie zal uitvoeren. Immers, als de persoon aan wie een verzoek wordt gericht met een belofte antwoordt, dan moet hij vervolgens wel de verlangde productieve actie uitvoeren. Aangezien de actorrol BA15 bepaalt aan wie alle coördinatieve acties gericht worden, en dus ook aan wie de verzoekende coördinatieve acties gericht worden, bepaalt hij daarmee wie welke productieve acties uitvoert. Over het algemeen zijn er in een organisatie veel personen die dezelfde actorrol kunnen spelen, en dus veel personen aan wie een coördinatieve actie gericht kan worden. Door een overwogen keuze te maken, kan de actorrol BA15 personen in een optimale verdeling toewijzen aan taken. Deze actorrol is bij uitstek met behulp van een scheduling-algoritme te beschrijven, en door een informatiesysteem te vervullen.

4.2.2 Het informatiesysteem levert informatie uit productiebanken

De actorrollen uit het bedrijfssysteem die niet door het informatiesysteem gespeeld worden, zullen door medewerkers van de organisatie gespeeld moeten worden. Het leveren van informatie aan actorrollen in het bedrijfssysteem is daarmee een tweede manier waarop het informatiesysteem ondersteuning kan bieden aan het bedrijfssysteem. Een aantal van deze actorrollen hebben voor het uitvoeren van hun taken namelijk informatie nodig, doordat er interstrictie plaatsvindt. Interstrictie houdt in dat actoren in hun handelen beperkt worden door het handelen van andere actoren. Daarom hebben actoren bij het uitvoeren van hun taken informatie nodig over het handelen van andere actoren. Deze informatiebehoefte leidt ertoe dat actoren soms willen weten wat er in andere transacties tot stand is gebracht; ze inspecteren daarom de productiebanken van andere transacties. Dit inspecteren van een productiebank is een informationele transactie. Er worden immers geen nieuwe feiten tot stand gebracht, maar informatie over reeds bestaande feiten doorgegeven. Net als iedere transactie hebben deze informationele transacties een actorrol die er de producer van is. Deze informationele actorrol zou door het informatiesysteem gespeeld kunnen worden.

Door alle actorrollen na te gaan die niet door het informatiesysteem worden gespeeld, en de informatiebehoefte daarvan te onderzoeken, kunnen we de informationele transacties vinden die zij initiëren. Daarmee kunnen we de eisen aan het informatiesysteem uitbreiden, door te stellen dat het informatiesysteem de executor van deze informationele transacties moet zijn. Eerst wordt de interstrictie besproken die voor elke actorrol tot een bepaalde informatiebehoefte leidt. Aan het einde van deze sectie is in tabel 4.2 een opsomming gegeven van de informatiebehoefte die uit deze interstrictie volgt.

Autorisatie De actorrol BA02 (die personen autoriseert voor bepaalde actorrollen) heeft informatie nodig over de transactieresultaten van T01, waarin alle personen zijn opgenomen die in de organisatie werkzaam zijn. Daarnaast heeft hij informatie uit drie externe feitenbanken nodig, waarin de transacties, proacties en actorrollen opgesomd zijn die in de organisatie voorkomen. De actorrol BA02 heeft een overzicht nodig

4.2. ONDERSTEUNING AAN HET BEDRIJFSSYSTEEM

van alle personen in de organisatie zodat hij weet aan wie hij eventueel een actorrol zou kunnen toekennen. Hij heeft uiteraard meer informatie nodig van die personen, bijvoorbeeld hun capaciteiten en loopbaanwensen, maar dat valt buiten het bestek van dit model. Hij heeft daarnaast informatie nodig over de actorrollen in de organisatie en de interacties waarin zij een rol spelen, zodat hij weet welke actorrollen door personen vervuld kunnen worden.

Agenda De actorrol BA04 selecteert uit de lijst van agenda een agendum om af te handelen. Deze actorrol heeft dus een lijst van voor hem geldende agenda nodig. Deze lijst is een deelverzameling van de transactieresultaten van transactie T08. Het resultaat van die transactie is namelijk een coördinatief feit. Niet alle coördinatieve feiten zijn agendum voor een actorrol, maar slechts die feiten waarop hij dient te reageren. De coördinatieve feiten waarin de actorrol niet voorkomt, noch als performer noch als adressee, zijn voor hem in ieder geval geen agendum. Indien de actorrol wel voorkomt in een coördinatief feit, dan is het afhankelijk van de intentie van dat coördinatieve feit of het een agendum vormt voor de actorrol. Een verzoek vormt bijvoorbeeld een agendum voor de adressee: aan hem is het gericht, en hij moet reageren met een belofte of een afwijzing. Maar een belofte daarentegen vormt een agendum voor de performer ervan. Een belofte verplicht de performer namelijk om een productieve actie te ondernemen en vervolgens te verklaren dat hij zijn belofte heeft ingelost. In tabel 4.1 staat een opsomming van alle intenties van de coördinatieve feiten (uit een transactie of een proactie) die een agendum vormen voor de adressee of de performer ervan.

De informatiebehoefte van actorrol BA04 bestaat uit een overzicht van alle agenda van een bepaalde actorrol A uit de organisatie. Zoals gezegd wordt actorrol BA04 gespeeld door een medewerker van de organisatie; het informatiesysteem kiest dus niet welk agendum afgehandeld moet worden, maar helpt bij het maken van die keuze door per actorrol een overzicht van al zijn agenda te tonen.

<i>agendum voor:</i>	<i>zijn alle feiten met intentie:</i>
adressee	request
”	state
”	decline
”	reject
performer	promise
adressee	advise
”	decline
performer	undertake

Tabel 4.1: Coördinatieve feiten vormen agendum voor performer of adressee

De actorrollen BA05 en BA06 hebben interstrictie met de externe feitenbank waarin de actieregels gedefinieerd zijn van alle actorrollen in de organisatie. Beide hebben informatie nodig over hoe de actieregels eruit zien: actorrol BA05 moet beslissen welke actieregel van toepassing is op het geselecteerde agendum, actorrol BA06 moet de actieregel toepassen. Maar, zoals eerder gezegd, maken we in dit onderzoek slechts gebruik van het coördinatiemodel en feitenmodel van een organisatie. Het actiemodel, waarin de actieregels gedefinieerd zijn, valt daar dus buiten. De informatiebehoefte van de actorrollen BA05 en BA06 kan dus niet door het informatiesysteem worden vervuld; het informatiesysteem heeft die informatie namelijk zelf ook niet.

Propositie De actorrol BA11 bepaalt de propositie van een (tot stand te brengen) coördinatief feit. Deze propositie beschrijft een toestand in de wereld waarover de coördinatieve actie een uitspraak doet. Een propositie is een bepaalde instantie van een propositietype, de propositie “Klacht 2355 is afgehandeld” bijvoorbeeld, is een instantie van het propositietype “Klacht K is afgehandeld”. Actorrol BA11 moet weten welke propositietypen horen bij welke transactietypen, en hoe de propositietypen gestructureerd zijn. Dit beperkt hem namelijk in zijn mogelijkheid om de propositie te bepalen van een coördinatief feit. De structuur van de propositietypen maakt deel uit van het DEMO-model van de organisatie, en is dus opgeslagen in een externe feitenbank. De actorrol BA11 heeft informatie uit die bank nodig.

Scheduling Ten aanzien van de actorrol BA15 hebben we opgemerkt dat die zal worden gespeeld door het informatiesysteem. De informatiebehoefte die voor deze actorrol geldt is daarom geen functionele eis aan het informatiesysteem.

4.2.3 Het informatiesysteem bewaart informatie

In een eenvoudige voorstelling van zaken houdt de executor van een transactie zelf de tot stand gebrachte transactieresultaten bij. Dit kan doordat elke essentiële actor ook een informationele en documentele actorrol omvat. We zeggen dat een essentiële actor een ‘informatieel jasje’ aan kan trekken. Als een andere actor informatie wil hebben over bepaalde transactieresultaten dan gaat hij een informationele transactie aan met de executerende actor van die transactie. Deze executor trekt daarvoor zijn informationele ‘jasje’ aan, voert de informationele transactie uit, en levert zo de gevraagde informatie.

In paragraaf 4.2.2 is beschreven dat sommige actorrollen informatie nodig hebben over tot stand gebrachte transactieresultaten, die het informatiesysteem zou moeten leveren. Deze actorrollen gaan dus niet een informationele transactie met de essentiële actorrol (in zijn informationele ‘jasje’) die deze transactieresultaten tot stand heeft gebracht; deze informationele actorrol wordt dan gespeeld door het informatiesysteem. Het informationele niveau wordt door de essentiële actor als het ware ‘uitbesteed’ aan het informatiesysteem, dat daarmee verantwoordelijk wordt voor het bewaren en aanbieden van informatie over de tot stand gebrachte transactieresultaten. Dit betekent dat het informatiesysteem de mogelijkheid biedt aan executors van transacties om voor hun informatie te bewaren over de tot stand gebrachte transactieresultaten. Tevens biedt het informatiesysteem aan actorrollen die daar behoefte aan hebben de mogelijkheid om die informatie op te vragen.

De vraag voor welke essentiële actoren het informatiesysteem de transactieresultaten moet bijhouden, is te beantwoorden door te kijken naar de vraag naar deze informatie. Immers, het bijhouden van transactieresultaten waar geen enkele actorrol om vraagt, is onnodig. Daarom is de informatiebehoefte gemodelleerd van actorrollen in het bedrijfssysteem. Het informatiesysteem speelt de informationele actorrollen die deze informatie, over de tot stand gebrachte transactieresultaten, bewaart en beschikbaar stelt. In tabel 4.2 is deze informatiebehoefte beschreven.

Voor actorrol BA08 geldt, dat hij voor het tot stand brengen van een coördinatief feit uiteraard moet weten wat er in de transacties BT11 t/m BT13 is bepaald. Actorrol BA15 heeft informatie nodig heeft om op een goede manier de personen in de organisatie in te plannen.

In tabel 4.3 staat de hiervan afgeleide behoefte die de essentiële actoren hebben om hun transactieresultaten te bewaren. Dit is zoals gezegd een informationele transac-

<i>actorrol</i>	<i>vraagt de transactieresultaten van:</i>
1. BA02:	BT01, de personen die in dienst zijn van de organisatie
2.	: EF1 , de transacties in deze organisatie
3.	: EF2 , de proacties in deze organisatie
4.	: EF3 , de actorrollen in deze organisatie
5. BA04:	BT08, alle voor een actorrol <i>A</i> geldende agenda
6.	: BT07, alle aan een actorrol <i>A</i> gedelegeerde agenda
7. BA06:	EF4 , de exit-points van de actieregels van een agendum
8. BA11:	EF5 , alle propositietypen in deze organisatie
9. BA08:	BT11, de propositie die voor dit coördinatieve feit is bepaald
10.	: BT12, het tijdsvenster dat voor dit coördinatieve feit is bepaald
11.	: BT13, de prioriteit die aan dit coördinatieve feit is gehecht
12.	: BT17, de bevestiging van dit coördinatieve feit
13. BA15:	BT02, de personen die geautoriseerd zijn voor een bepaalde actorrol
14.	: BT03, de tijd die personen voor een actorrol beschikbaar hebben
15.	: BT09, het verlof dat personen hebben gekregen
16.	: BT10, de geschatte van een productieve actie
17. BA17:	BT16, het geïnformeerd zijn van een coördinatieve actie

Tabel 4.2: Informatiebehoefte van actorrollen uit het bedrijfssysteem, vloeit voort uit de interstructie

tie, waarbij de essentiële actoren het informatiesysteem verzoeken bepaalde informatie voor hen te bewaren. In tabel 4.4 staat een totaaloverzicht gegeven van de gehele informatiebehoefte.

4.3 Functioneel ontwerp

In de voorgaande sectie zijn de functionele eisen opgesteld die we stellen aan het informatiesysteem. De functionaliteit van het informatiesysteem bestaat enerzijds uit het vervullen van een aantal actorrollen uit het bedrijfssysteem en anderzijds uit het leveren van informatie aan gebruikers, die zelf actorrollen uit het bedrijfssysteem spelen. Om dit te kunnen doen, moet het informatiesysteem op de hoogte worden gebracht van transactieresultaten die buiten het informatiesysteem tot stand zijn gekomen.

Het functionele ontwerp van het informatiesysteem bestaat uit de verzameling van de hiervoor genoemde functionele eisen. Het functionele ontwerp beschrijft de functionaliteit die het informatiesysteem aan de gebruikers moet aanbieden, en gaat niet in op de manier waarop deze functionaliteit gerealiseerd moet worden. Een functioneel ontwerp is dus op de buitenkant van het informatiesysteem gericht: wat moet het informatiesysteem aan de buitenwereld aanbieden.

Een functioneel ontwerp kan in verschillende modelleertechnieken weergegeven worden. Een veelgebruikte modelleertechniek voor informatiesystemen is de Unified Modeling Language. In UML kan een functioneel ontwerp bijvoorbeeld met behulp van Use Cases beschreven worden: deze geven aan wat de gebruiksmogelijkheden van het informatiesysteem zijn.

Een functionele eis beschrijft een dienst die de ene partij aan de andere partij moet leveren; het gaat over het tot stand brengen van een verandering in een toestandswereld. Een DEMO-transactie beschrijft precies dit: de verplichting van de ene partij om op

HOOFDSTUK 4. FUNCTIONEEL ONTWERP VAN EEN WFM SYSTEEM

<i>actorrol</i>	<i>vraagt het bewaren van de transactieresultaten van:</i>	<i>voor:</i>
18. BA01:	BT01, de personen die in dienst zijn bij de organisatie	1.
19. BA02:	BT02, de personen die geautoriseerd zijn voor een actorrol	13.
20. BA03:	BT03, de tijd die personen voor een actorrol beschikbaar hebben	14.
21. BA07:	BT07, de agenda die aan personen zijn gedelegeerd	6.
22. BA09:	BT09, het verlof dat personen hebben gekregen	15.
23. BA10:	BT10, de geschatte tijdsduur van een productieve actie	16.
24. BA11:	BT11, de propositie van een coördinatief feit	9.
25. BA12:	BT12, het tijdsvenster van een coördinatief feit	10.
26. BA13:	BT13, de prioriteit van een coördinatief feit	11.
27. BA17:	BT17, de bevestiging van een coördinatief feit	12.
28. BA16:	BT16, het geïnformeerd zijn een coördinatief feit	17.

Tabel 4.3: Behoeft tot het bewaren van informatie, vloeit voort uit informatiebehoefte uit tabel 4.2

<i>Actorrol</i>	<i>Informatiebehoefte:</i>
1. BA02:	BT01, de personen die in dienst zijn van de organisatie
2.	: EF1 , de transacties in deze organisatie
3.	: EF2 , de proacties in deze organisatie
4.	: EF3 , de actorrollen in deze organisatie
5. BA04:	BT08, alle voor een actorrol A geldende agenda
6.	: BT07, alle aan een actorrol A gedelegeerde agenda
7. BA06:	EF4 , de exit-points van de actieregels van een agendum
8. BA11:	EF5 , alle propositietypen in deze organisatie
9. BA08:	BT11, de propositie die voor dit coördinatieve feit is bepaald
10.	: BT12, het tijdsvenster dat voor dit coördinatieve feit is bepaald
11.	: BT13, de prioriteit die aan dit coördinatieve feit is gehecht
12.	: BT17, de bevestiging van dit coördinatieve feit
13. BA15:	BT02, de personen die geautoriseerd zijn voor een bepaalde actorrol
14.	: BT03, de tijd die personen voor een actorrol beschikbaar hebben
15.	: BT09, het verlof dat personen hebben gekregen
16.	: BT10, de geschatte tijdsduur van een productieve actie
17. BA17:	BT16, het geïnformeerd zijn van een coördinatieve actie
18. BA01:	BT01, de personen die in dienst zijn bij de organisatie
19. BA02:	BT02, de personen die geautoriseerd zijn voor een actorrol
20. BA03:	BT03, de tijd die personen voor een actorrol beschikbaar hebben
21. BA07:	BT07, de agenda die aan personen zijn gedelegeerd
22. BA09:	BT09, het verlof dat personen hebben gekregen
23. BA10:	BT10, de geschatte tijdsduur van een productieve actie
24. BA11:	BT11, de propositie van een coördinatief feit
25. BA12:	BT12, het tijdsvenster van een coördinatief feit
26. BA13:	BT13, de prioriteit van een coördinatief feit
27. BA17:	BT17, de bevestiging van een coördinatief feit
28. BA16:	BT16, het geïnformeerd zijn een coördinatief feit

Tabel 4.4: Totale informatiebehoefte

verzoek van de andere partij een toestandsverandering tot stand te brengen. DEMO-transacties zijn daarom een -andere- manier om functionele eisen op te stellen.

In dit onderzoek is ervoor gekozen om het functionele ontwerp in een DEMO-model op te stellen. Een reden hiervoor is dat op deze manier de relaties tussen verschillende modellen duidelijk blijven: ten eerste de relatie tussen het bedrijfssysteem en het informatiesysteem, en ten tweede de relatie tussen het functioneel ontwerp en de informatiestructuur. Het functionele ontwerp is zonder veel moeite naar Use Cases om te zetten, en had ook in UML gemaakt kunnen worden; de keuze om dit in DEMO te doen is ingegeven door de wens in het onderzoek één lijn vast te houden en daarmee het verband tussen bedrijfssysteem, informatiesysteem en informatiestructuur zichtbaar te maken.

In figuur 4.1 is het functionele ontwerp van het informatiesysteem weergegeven. Weergegeven zijn alle informatiele en essentiële transacties waarvan het informatiesysteem executor is. De functionaliteit van het informatiesysteem bestaat eruit dat het al deze transacties uit kan voeren. In tabel 4.5 is beschreven hoe deze functionaliteit is te herleiden tot de in tabel 4.4 beschreven informatiebehoefte van actorrollen. Alle functionele eisen zijn te herleiden tot óf een informatiebehoefte van één van de actorrollen uit het bedrijfssysteem, óf het vervullen van een actorrol uit het bedrijfssysteem.

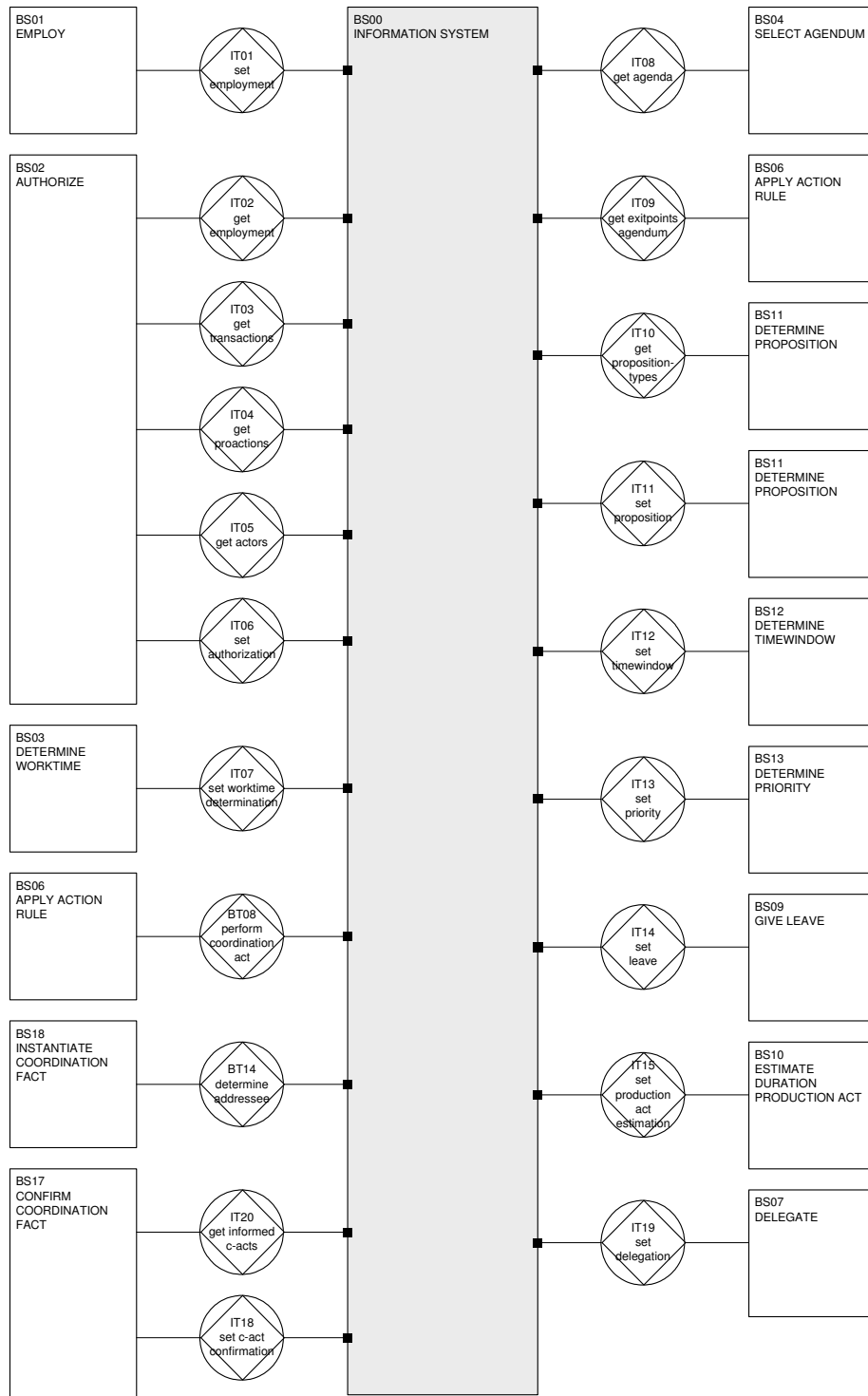
<i>door informatiesysteem aangeboden</i>	<i>afkomstig uit informatiebehoefte</i>
IT01 : set employment	17: BA01 <i>bewaar</i> transactieresultaten BT01
IT02 : get employment	1: BA02 <i>vertel</i> de personen in de organisatie
IT03 : get transactions	2: BA02 <i>vertel</i> alle transacties in organisatie
IT04 : get proactions	3: BA02 <i>vertel</i> alle proacties in organisatie
IT05 : get actors	4: BA02 <i>vertel</i> alle actorrollen in organisatie
IT06 : set authorization	18: BA02 <i>bewaar</i> transactieresultaten BT02
IT07 : set worktime determination	19: BA03 <i>bewaar</i> transactieresultaten BT03
IT08 : get agenda	5: BA04 <i>vertel</i> alle agenda voor actorrol A
IT09 : get exitpoints agendum	7: BA06 <i>vertel</i> exit-points van een agendum
IT10 : get proposition-types	8: BA11 <i>vertel</i> propositietypen organisatie
IT11 : set proposition	23: BA11 <i>bewaar</i> transactieresultaten BT11
IT12 : set timewindow	24: BA12 <i>bewaar</i> transactieresultaten BT12
IT13 : set priority	25: BA13 <i>bewaar</i> transactieresultaten BT13
IT14 : set leave	21: BA09 <i>bewaar</i> transactieresultaten BT09
IT15 : set production act estimation	22: BA10 <i>bewaar</i> transactieresultaten BT10
BT16: perform coordination act	<i>executor</i> BA08
BT17: determine addressee	<i>executor</i> BA18
IT18 : set confirmation	26: BA17 <i>bewaar</i> transactieresultaten BT17
IT19 : set delegation	20: BA07 <i>bewaar</i> transactieresultaten BT07
IT20 : get informed c-acts	27: BA16 <i>bewaar</i> BT16

Tabel 4.5: Verband functionaliteit informatiesysteem en informatiebehoefte actorrollen

4.4 Toetsing aan functionele eisen wfm systemen

Het functionele ontwerp van het informatiesysteem is volledig uit de CAP-theorie afgeleid. Eerst zijn de theoretische uitgangspunten van CAP over de werking van or-

HOOFDSTUK 4. FUNCTIONEEL ONTWERP VAN EEN WFM SYSTEEM



Figuur 4.1: Functioneel ontwerp informatiesysteem

ganisaties vertaald naar een bedrijfssysteem dat beschrijft hoe werknemers hun taken selecteren, uitvoeren en met elkaar samenwerken. Vervolgens is een informatiesysteem beschreven dat dit bedrijfssysteem ondersteunt. De eisen aan dit informatiesysteem zijn volledig afkomstig uit de beschrijving van het bedrijfssysteem; en het bedrijfssysteem is volledig afkomstig uit de CAP-theorie.

Het zou daarom dogmatisch zijn om te stellen dat het functioneel ontwerp voldoet aan de informatiebehoefte van het bedrijfssysteem, en daarmee een goed *wfm* systeem beschrijft, aangezien het hele informatiesysteem vanuit één uitgangspunt is opgesteld. De functionaliteit van het ontworpen informatiesysteem is alleen te beoordelen door buiten het kader van de CAP-theorie te treden, en het functionele ontwerp van daar te bekijken.

Daarom is in hoofdstuk 2 een algemene beschrijving gegeven van *wfm* systemen. Hierbij is een overzicht gegeven van de belangrijkste uitgangspunten in de literatuur over *wfm* systemen. Aan de hand van definities van de Workflow Management Coalition zijn algemene functionele eisen opgesteld, waaraan een *wfm* systeem zou moeten voldoen. Om te controleren of het opgestelde functionele ontwerp als een *wfm* systeem kan worden gekwalificeerd, zullen we dit ontwerp toetsen aan de functionele eisen uit hoofdstuk 2. Deze eisen houden geen verband met de DEMO-theorie en zijn daardoor een goede graadmeter voor ons functionele ontwerp.

Helaas zijn de definities van de Coalition en de functionele eisen die daarvan afgeleid zijn niet erg scherp. Toetsing aan deze functionele eisen levert daarom niet erg veel op; doordat de eisen erg breed zijn, voldoet een informatiesysteem er al snel aan. De reden hiervoor is dat er, zoals in hoofdstuk 2 is beschreven, weinig overeenstemming bestaat over de benodigde functionaliteit van *wfm* systemen. Het is echter belangrijk om te weten hoe dit ontwerp, dat volledig afkomstig is uit de CAP-theorie en dus communicatie-gebaseerd is, zich verhoudt tot andere *wfm* systemen, die zich over het algemeen sterk richten op de activiteiten in een organisatie. Daarom vergelijken we ons functionele ontwerp ook met de functionaliteit die andere *wfm* systemen bieden.

4.4.1 Vergelijking met WfMC eisen

In paragraaf 2.4 is beschreven hoe volgens de Workflow Management Coalition een *wfm* systeem eruit moet zien. Deze beschrijving gaat ten eerste in op de architectuur van een *wfm* systeem, dat wil zeggen de componenten waaruit het moet bestaan en de structuur en inhoud van de interfaces die het moet aanbieden aan andere systemen. Ten tweede zegt deze beschrijving iets over de functionaliteit die een *wfm* systeem aan zijn gebruikers moet aanbieden.

De door hun beschreven architectuur van een *wfm* systeem, die uit een aantal componenten moet bestaan die op een bepaalde manier met elkaar samenwerken, is voor ons niet relevant. De functionaliteit die deze componenten moeten aanbieden echter wel; de som van de functionaliteit van de componenten vormt immers de gewenste functionaliteit van het *wfm* systeem.

De Coalition onderscheidt drie componenten, die ieder een bepaalde functionaliteit moet aanbieden.

- Ten eerste noemt de Coalition een *modelleergereedschap*, waarmee de definitie van het bedrijfsproces kan worden opgesteld. De functionaliteit om bedrijfsprocessen te modelleren wordt door dit informatiesysteem (de ActieManager) niet zelf aangeboden, maar is aanwezig in de vorm van de reeds bestaande Designer. Dit is een softwarepakket van ModulOr Technology waarmee DEMO-modellen

van een organisatie zijn op te stellen. De ActieManager heeft deze modellen uiteraard nodig, en zonder de Designer is de ActieManager dan ook geen volwaardig *wfm* systeem te noemen. Het uitgangspunt van dit onderzoek is ook steeds geweest dat deze twee producten op deze manier met elkaar zouden samenwerken.

- Ten tweede noemt de Coalition een *worklist handler*, waarmee de interactie tussen de gebruikers en hun werklis t beheerd wordt. Aan deze component worden twee functionele eisen gesteld:
 - *het biedt aan het wfm systeem de mogelijkheid om taken aan gebruikers toe te wijzen*: Dit wordt door de ActieManager aangeboden in de vorm van de transactie IT08, *get agenda*. Deze transactie geeft de gebruiker de mogelijkheid om zijn agenda, zijn werklis t, op te vragen.
 - *het biedt aan gebruikers de mogelijkheid bieden om statuswijzigingen in de taken aan het systeem door te geven*: Dit wordt door de ActieManager aangeboden in de vorm van de transactie BT16, *perform coordination act*. Door het uitvoeren van een coördinatieve actie brengt een actor namelijk een wijziging aan in de status van een taak, de precieze wijziging hangt af van de inhoud van de coördinatieve actie.
- Ten derde noemt de Coalition een *workflow engine*, de executie omgeving voor instanties van het bedrijfsproces. De workflow engine biedt de volgende functionaliteit:
 - *het interpreteert de bedrijfsprocesdefinitie*: dit is geen functionaliteit, maar is noodzakelijk om de het toewijzen van taken aan resources te kunnen uitvoeren. De ActieManager doet dit in de transacties IT03 t/m IT05.
 - *het beheert de verschillende instanties van het bedrijfsproces en navigeert ertussen*: dit betekent dat het *wfm* systeem bijhoudt welke instanties van het bedrijfsproces er zijn, en ervoor zorgt dat deze allemaal afgehandeld worden.

Het beheren van de verschillende instanties van het bedrijfsproces moet uiteraard gebeuren, maar is geen functionele eis. Het navigeren tussen de verschillende instanties en ervoor zorgen dat deze allemaal afgehandeld worden is in de ActieManager de verantwoordelijkheid van de gebruikers. Een gebruiker krijgt een overzicht van de verschillende instanties van het bedrijfsproces, de verschillende instanties van transacties, die voor hem van belang zijn. Het is zijn eigen verantwoordelijkheid, samen met de partij waarmee hij een transactie aangaat, om zijn beloften na te komen.
 - *het wijst taken toe aan resources*: dit doet de ActieManager in transactie BT17, *determine addressee*. Het bepalen van een addressee van een verzoek, staat daarmee gelijk aan het toewijzen van een taak aan die addressee. Uiteraard staat het de addressee vrij om het verzoek te weigeren, maar dit zal hij slechts doen in overeenstemming met de bedrijfsregels.

De ActieManager wijst zelf geen taken toe aan resources, maar bepaalt, of doet een voorstel voor, de addressee van een coördinatieve actie. Pas als een addressee van een verzoek reageert met een belofte, heeft hij die taak op zich genomen. In die zin neemt elke resource ‘vrijwillig’ taken op zich, en worden ze niet toegewezen. Maar actoren zijn niet vrij om zomaar

verzoeken te beloven of te weigeren, ze doen dat in overeenstemming met hun actieregels, die de bedrijfsregels weerspiegelen. Vrijwillig zijn actoren dus niet in het beloven of afwijzen.

Na het bepalen van de adressee van een verzoekende coördinatieve actie, volgt het verzoek aan diegene om een propositie tot stand te brengen. Door het beloven daarvan, neemt de adressee de taak op zich. En aangezien het beloven of weigeren niet vrijwillig plaatsvindt, maar als uiting van afspraken binnen de organisatie, houdt het bepalen van een adressee van een verzoekende coördinatieve actie het toewijzen van die adressee aan die taak in, en biedt de ActieManager deze functionaliteit dus in transactie BT17, *determine adressee*.

- *het meldt gebruikers aan/af*: het aan- en afmelden van gebruikers is noodzakelijk, zodat het *wfm* systeem weet welke gebruikers er beschikbaar zijn. Hiervoor moet zowel bekend zijn welke personen er werkzaam zijn in de organisatie en welke actorrollen zij kunnen vervullen. Daarnaast moet bekend zijn hoeveel tijd zij daarvoor in het algemeen beschikbaar hebben en, op een operationele basis, de aan- en afwezigheid op de werkvloer (in geval van ziekte, vakantie, etc.).

De mogelijkheid om het indiensttreden en autoriseren voor één of meer actorrollen aan het systeem door te geven biedt de ActieManager in de transacties IT01, *set employment* en IT06, *set authorization*. Hierin zit ook de mogelijkheid gemodelleerd om uitdiensttredingen en het afnemen van autorisaties door te geven.

De mogelijkheid om de beschikbaarheid, zowel op lange termijn als op (incidentele) korte termijn, aan het systeem door te geven, biedt de ActieManager in de transacties IT07, *set worktime determination* en IT14, *set leave*.

- *het biedt een interface aan externe applicaties en workflow engines*: dit is een functionaliteit die de ActieManager niet aanbiedt. Dit komt doordat het ten eerste tijdens dit onderzoek niet haalbaar was om te implementeren en daarom vanaf het begin van het project niet in het functionele ontwerp is meegenomen.

Ten tweede komt dit doordat deze functionaliteit voor de beantwoording van de onderzoeksvraag niet noodzakelijk is. Koppelingen met externe systemen en (in mindere mate) met externe workflow engines zijn voor de praktische implementatie van een *wfm* systeem weliswaar haast onontbeerlijk, maar voegen niets toe aan de mogelijkheid om de bedrijfsprocessen te sturen en taken aan gebruikers toe te wijzen.

Daarbij is het altijd mogelijk om een interface aan externe applicaties aan te bieden in een later stadium, bij het maken van een commercieel product.

- *het biedt (een interface tot) toezicht mogelijkheden*: dit is een functionaliteit die de ActieManager niet aanbiedt. Hiervoor gelden precies dezelfde redenen als voor bovenstaand punt: het was niet haalbaar in het project, en niet noodzakelijk voor beantwoording van de onderzoeksvraag. Daarbij is het zonder problemen in een commercieel product aan te bieden.

Ten aanzien van de functionaliteit die de Workflow Management Coalition voorschrijft aan *wfm* systemen kunnen we vaststellen dat het getoonde functionele ontwerp

van de ActieManager daar, met uitzondering van twee punten, aan voldoet. De functionaliteit die de ActieManager niet aanbiedt (koppelingen met andere systemen en toezichtmogelijkheden) zijn niet essentieel voor het beantwoorden van de onderzoeksvraag en in een later stadium toe te voegen. Het aanbieden van deze functionaliteit is zeker mogelijk, maar wegens beperkingen in tijd niet gedaan.

4.4.2 Vergelijking met andere *wfm* systemen

Helaas zijn de definities van de Coalition en de functionele eisen die daarvan afgeleid zijn niet erg scherp. Om te weten hoe de ActieManager zich verhoudt tot andere *wfm* systemen, vergelijken we ons functionele ontwerp met de functionaliteit van drie andere *wfm* systemen: Staffware, COSA en ActionWorkflow. De beschrijving van de functionaliteit van deze systemen is afkomstig uit de literatuur; helaas was er geen mogelijkheid deze te bekijken.

Staffware Staffware is de marktleider in *wfm* systemen, met ongeveer een kwart van de wereldmarkt. Staffware bestaat uit een aantal componenten [4]: een gereedschap om processen te definiëren met gebruik van een eigen modelleertechniek; een component om de user interface die gebruikers te zien krijgen te definiëren; een werklis manager waarmee gebruikers met het systeem kunnen communiceren; een workflow engine; een component waarmee de voortgang van individuele cases is te volgen; en tenslotte een component om het systeem te beheren.

In vergelijking met de ActieManager biedt Staffware qua functionaliteit duidelijk een stuk meer: de mogelijkheid om de user interface aan te passen aan de eigen wensen, de mogelijkheid om individuele cases te volgen en ten dele de mogelijkheden voor systeembeheer worden door de ActieManager niet geboden.

Een belangrijk verschil is ook dat Staffware voor het definiëren van de bedrijfsprocessen gebruik maakt van een eigen modelleertechniek. Deze is ten dele een subset van de Petri-net techniek, biedt dus minder mogelijkheden, en verschilt er op een aantal concepten van. Daardoor kunnen modellen, in vergelijking met hoog-niveau Petri-netten, groot en complex worden en zijn sommige constructies niet te modelleren [4].

COSA COSA is een op Petri-netten gebaseerd *wfm* systeem. COSA bestaat uit een aantal componenten [4]: een gereedschap om de definities van bedrijfsprocessen op te stellen, gebruikmakend van Petri-netten; een user editor waarmee resources zijn te structureren in rollen in organisatorische eenheden; een werklis handler waarmee elke medewerker met het systeem interacteert; een gereedschap dat de status van een individuele case kan tonen; een workflow engine; een simulatie-gereedschap; en een component voor systeembeheer. COSA biedt ook de mogelijkheid aan gebruikers om hun werklis via het internet te benaderen, zodat ze overal hun taken kunnen zien, en het door hun gedane werk kunnen aangeven.

COSA is volledig op Petri-netten gebaseerd. Het voordeel daarvan is dat er veel wetenschappelijke methoden zijn ontwikkeld om Petri-netten te analyseren en (kwantitatief) te optimaliseren. Zo bestaat er een gereedschap Woflan [26] dat, door middel van allerlei wiskundige technieken, een uitspraak kan doen over een bedrijfsprocesdefinitie. Woflan kan een model vanuit COSA inlezen en analyseren. Vervolgens kan het fouten of problemen opsporen en oplossingen aanreiken om deze te verhelpen. Woflan

kan ook modellen uit Staffware inlezen, aangezien Staffware een modelleertechniek gebruikt die lijkt op een subset van Petri-netten.

Ook zijn Petri-netten zeer goed te simuleren en (kwantitatief) te optimaliseren, bijvoorbeeld met het simulatiegereedschap ExSpect [9]. ExSpect kan bedrijfsprocessen uit onder andere COSA inlezen en simuleren, zodat knelpunten kunnen worden opgespoord. Uit simulaties kunnen allerlei kentallen worden berekend, zoals wachttijden, doorvoertijden en bezettingsgraad, waarmee kwantitatieve uitspraken over het bedrijfsproces kunnen worden gedaan [3]. Ook biedt ExSpect de mogelijkheid om veranderingen in het bedrijfsproces ‘op papier’ uit te proberen, en door simulatie het verwachte resultaat te berekenen.

In vergelijking met de ActieManager biedt COSA wat meer functionaliteit aan. Het volgen van individuele casussen kan in de ActieManager niet, en een simulatiegereedschap biedt de ActieManager ook niet. Een groot verschil is dat COSA, door het gebruik van Petri-netten, goede mogelijkheden biedt tot samenwerking met andere pakketten. Deze pakketten maken onder andere gebruik van de stevige wetenschappelijke basis voor het analyseren en optimaliseren van Petri-netten. COSA biedt zelf deze functionaliteit niet aan, maar maakt het voor gebruikers wel aantrekkelijk in combinatie met zulke verificatie- en optimalisatie-pakketten te gebruiken.

ActionWorkflow ActionWorkflow is een communicatie-gebaseerd *wfm* systeem. ActionWorkflow beschouwt, net als DEMO, bedrijfsprocessen vanuit het Language/Action paradigma. ActionWorkflow onderscheidt in een bedrijfsproces, en de ondersteuning daarvan door een informatiesysteem, de volgende concepten [23]:

- *language acts*, vergelijkbaar met coördinatieve acties in DEMO. In overeenstemming met de speech-act theorie hanteert ActionWorkflow het uitgangspunt dat communicatie, in de vorm van natuurlijke taal, opgevat kan worden als aanzetten tot of het doen van een actie.
- *conversations*, een samenhangende opeenvolging van language acts met een vastgestelde structuur.
- *time tokens*, die een tijdsduur aangeven waarin een conversatie afgerond moet zijn.
- *workflow loops*, een patroon van acties waarin de customer zijn verwachtingen en wensen uit en de performer deze tot stand brengt. Een workflow loop is vergelijkbaar met een transactie in DEMO. Een workflow loop bestaat uit vier fasen waarin wordt overlegd en afspraken worden gemaakt. Een workflow loop begint met de *proposal* fase, waarin de customer iets verzoekt aan de performer. Het is ook mogelijk dat de performer juist aan de customer aanbiedt om iets tot stand te brengen. In de *agreement* fase spreken de beide partijen de voorwaarden voor acceptatie af, waaronder de tijdsduur van de verdere stappen. In de *performance* fase zorgt de performer voor het uitvoeren van de gewenste handeling, en verklaart aan de customer dit te hebben gedaan. In de *satisfaction* fase verklaart de customer tenslotte aan de performer tevreden te zijn met de afhandeling van het verzoek.

ActionWorkflow beschouwt een bedrijfsproces als een samenstelling (“interweaving”) van allerlei workflow loops. Vanuit elke fase in een workflow loop kunnen één

of meerdere andere workflow loops opgestart worden. Ook kunnen fasen van verschillende workflow loops wachtcondities voor elkaar vormen. Juist omdat de denkwijze van ActionWorkflow lijkt op die van DEMO, is het interessant om de functionaliteit van het product te bekijken:

ActionWorkflow bestaat uit een aantal componenten [4]: een gereedschap voor het definiëren van de workflows in het bedrijfsproces; een procesmanager, die zowel een workflow engine is, als een gereedschap om de workflow te beheren en te analyseren; een component die integratie met document beheer systemen vergemakkelijkt; en een component waarmee gebruikers hun worklist via het internet kunnen beheren.

ActionWorkflow gaat uit van soortgelijke principes als de ActieManager. In functionaliteit is het breder dan de ActieManager, het beheren en analyseren van de actieve workflows is in de ActieManager niet mogelijk. Ook de mogelijke koppeling met document beheer systemen is een functionaliteit die de ActieManager niet aanbiedt, en in de praktijk zeer handig kan zijn.

Trends In [4] worden een aantal aspecten opgenoemd van *wfm* systemen die naar verwachting een (gaan) rol spelen in de toekomst. Deze aspecten worden benaderd als kansen of bedreigingen voor de invoering en acceptatie van *wfm* systemen door organisaties en gebruikers. Deze aspecten, en de verwachte trends daarin, zijn ook een indicatie van de gewenste functionaliteit van *wfm* systemen. We proberen, door de ActieManager tegen het licht te houden ten aanzien van deze aspecten, een beeld te vormen van de eigen functionaliteit in verhouding met andere systemen en (verwachte) marktvrage.

- *modelleren*: het modelleren van bedrijfsprocessen is de basis van een *wfm* systeem. Veel van de huidige *wfm* systemen zijn niet gebaseerd op goede procesmodellen [4]. Hierdoor is het niet mogelijk om veel voorkomende constructies in bedrijfsprocessen te modelleren, en te ondersteunen. In [4] wordt de verwachting uitgesproken dat in de toekomst alle *wfm* systemen bedrijfsprocessen met behulp van Petri-netten zullen modelleren, aangezien deze modelleertechniek dit probleem niet kent.

In [5] worden 20 workflow patronen beschreven, standaard constructies in bedrijfsprocessen. Het uitgangspunt is dat dit basispatronen zijn die vaak voorkomen in bedrijfsprocessen. Ondersteuning ervan is daarom noodzakelijk voor het aanbieden van een goede, veelomvattende *wfm* functionaliteit.

Vervolgens wordt beschreven in hoeverre op de markt beschikbare *wfm* systemen deze patronen ondersteunen. De conclusie is dat van veel pakketten een dusdanige modelleertechnieken gebruiken, dat slechts de eenvoudige patronen ermee te modelleren zijn. Dit hoeft geen reden te zijn om het pakket niet te gebruiken, maar zegt wel iets over de gebruikte modelleertechniek.

In [10] wordt beschreven hoe in DEMO deze patronen ondersteund worden. Ondanks het feit dat DEMO een totaal andere benadering volgt voor het beschrijven van bedrijfsprocessen, is het volgens [10] mogelijk om in de actieregels van een actor al deze patronen te modelleren. Dit is nog niet door andere bronnen bevestigd, maar toont wel aan dat DEMO, en daarmee de ActieManager, op modelleerkracht de vergelijking met andere pakketten en hun technieken kan doorstaan.

Daarnaast valt zowel over de beschreven patronen, als over de opmerking dat in de toekomst alle *wfm* systemen gebaseerd zullen zijn op Petri-netten omdat deze als enige deze patronen kunnen modelleren, te discussiëren. DEMO gebruikt

een subset van Petri-netten in het processtapdiagram, maar stelt dat het beschouwen van organisaties en hun bedrijfsprocessen vanuit de (fysieke) activiteiten die plaatsvinden, een verkeerde aanpak is. Een verkeerd uitgangspunt, dat daardoor leidt tot een verkeerd beeld van de organisatie.

- *analyseren*: Het analyseren van bedrijfsprocessen, zowel kwalitatief (correctheid, werkbaarheid, etc.) als kwantitatief (doorlooptijden, benuttingpercentages, wachttijden etc.) is een belangrijke functionaliteit. Dit kan zowel worden ingezet voor het analyseren van bestaande bedrijfsprocessen, als het beoordelen van geplande veranderingen. Ook simulatie speelt daarbij een rol, vooral in combinatie met zogeheten ‘management-games’, waarbij mensen met behulp van rollenspellen het werk in de organisatie naspelen, met de bedoeling om zwakke plekken en veranderingsmogelijkheden te vinden.

Gereedschappen en technieken hiervoor bestaan indien de bedrijfsprocessen met behulp van Petri-netten zijn opgesteld. Dit is voor DEMO-modellen niet het geval; er zijn (op dit moment) geen gereedschappen beschikbaar voor verificatie of analyse van DEMO-modellen.

- *planning*: de huidige generatie *wfm* systemen besteedt niet veel aandacht aan het, op operationeel niveau, optimaal inplannen van menselijke resources, hetgeen wel belangrijk is. Technieken op de gebieden van operations research en kunstmatige intelligentie zijn hier wel sterk in. Daarnaast zou een *wfm* systeem in principe ook de informatie moeten bevatten om tactische planning, de schatting van de verwachte benodigde capaciteit van resources, te ondersteunen.

Noch op operationeel gebied, noch op tactisch gebied, bieden huidige *wfm* systemen goede ondersteuning. Ook de ActieManager heeft slechts een rudimentaire vorm van scheduling ingebouwd. Het gemis aan deze functionaliteit is echter, zowel bij commerciële systemen als de ActieManager, niet het gevolg van een fundamenteel probleem, maar een te magere technische implementatie. Juist door technieken en algoritmen te gebruiken uit voorgenoemde gebieden, zouden *wfm* systemen in de toekomst deze planning moeten kunnen uitvoeren.

- *transactie management*: *wfm* systemen beperken zich over het algemeen tot bedrijfsprocessen binnen één organisatie. Hierbij gaan ze ervan uit dat ten eerste alle resources slechts voor die ene organisatie werken, ten tweede dat de organisatie volledig kan bepalen welke taken de resources uitvoeren, en ten derde dat alle resources dezelfde informatiesystemen en communicatiemiddelen gebruiken. Deze veronderstellingen zijn niet problematisch als het *wfm* systeem zich inderdaad beperkt tot één organisatie.

Maar deze uitgangspunten zijn niet geldig als het *wfm* systeem bedrijfsprocessen moet ondersteunen die over de organisatiegrenzen heen reiken, bijvoorbeeld bij netwerkorganisaties zonder vaste medewerkers, of bij elektronische transacties tussen verschillende organisaties.

Daarom is er een behoefte aan *wfm* systemen die niet zomaar taken aan resources toewijzen, maar relevante resources kunnen selecteren, en vervolgens met die resources te onderhandelen over de taken die zij zullen gaan uitvoeren; daarnaast is er behoefte aan *wfm* systemen die elektronische samenwerking mogelijk maken tussen organisaties, bijvoorbeeld met behulp van XML.

De ActieManager biedt voor dit laatste geen ondersteuning. De ActieManager is in het geheel niet gericht op samenwerking met andere systemen, en al helemaal niet op de manier van communiceren daarbij. Maar DEMO biedt op dit gebied wel perspectieven. In [11] is beschreven hoe, met behulp van het nieuwe *netwerkmodel*, de samenwerking tussen verschillende partijen kan worden beschreven. Met behulp van deze aanpak is het vervolgens mogelijk om voor elk samenwerkingsverband afspraken op te stellen over hetgeen aan elkaar geleverd wordt, de informatie die uitgewisseld moet worden en de vorm waarin deze informatie-uitwisseling moet plaatsvinden.

- *interoperabiliteit*: *wfm* systemen voeren zelf geen werk uit, maar verdelen slechts het werk over resources. Veel van dit werk wordt niet door mensen gedaan, maar door andere informatiesystemen. Het *wfm* systeem zou idealiter ook opdrachten moeten kunnen geven aan deze informatiesystemen, in plaats van tegen een mens te zeggen dat hij het informatiesysteem iets moet laten doen. In [4] wordt een toekomstbeeld geschetst waarin een *wfm* systeem als een besturingssysteem alle informatiestromen in de organisatie beheert, en alle informatieverwerkers, mensen en computers, aanstuurt. Hiervoor zullen de interfaces van *wfm* systemen sterk uitgebreid en gestandaardiseerd moeten worden.
- *internet*: een toenemend aantal *wfm* systemen biedt aan gebruikers de mogelijkheid om hun werklis te benaderen via het internet. Het voordeel daarvan is ten eerste dat zij van elke plek met een internet-verbinding bij hun werklis kunnen. Het tweede voordeel is dat er geen client-software nodig is, behalve de standaard aanwezige internet browser. Een aanvullend voordeel bestaat uit de koppeling van een *wfm* systeem aan een e-commerce applicatie, zoals een elektronische boekwinkel. Een door een klant geplaatste order kan dan automatisch in de werklis van een medewerker terechtkomen.

Hoewel dit strikt gezien geen uitbreiding is van de functionaliteit van een *wfm* systeem, is het wel een belangrijk punt. Tijdens de bespreking van het prototype, in het volgende hoofdstuk, zullen we zien dat daar is gekozen voor het beschikbaar maken van de werklis via het internet, precies om bovengenoemde redenen.

- *logistiek management*: logistieke systemen, of ERP-systemen, zijn succesvol en wijd verspreid in organisaties. Deze systemen bieden ondersteuning bij de financiële administratie, bij voorraadbeheer, en voor communicatie en afspraken met leveranciers en klanten. Deze systemen kunnen onder andere uit een “bill-of-material” van een bestelling, automatisch de daarvoor benodigde orders plaatsen bij leveranciers en het werk plannen dat moet worden uitgevoerd.

Wfm systemen zouden een deel van deze functionaliteit kunnen overnemen, en aanpassen aan het eigen werkterrein. De mogelijkheid om uit een “bill-of-material” automatisch het werk af te leiden dat plaats moet vinden, zou in een *wfm* systeem ook gebruikt kunnen worden. Er is daarbij geen sprake van een “bill-of-material”, maar een soort “bill-of-information”. Door te bepalen welke informatie er nodig is voor het afhandelen van een taak, kan het *wfm* systeem ten eerste de taken afleiden die gedaan moeten worden, en daarmee ondersteuning bieden bij het modelleren van het bedrijfsproces. Ten tweede zou het *wfm* systeem alle benodigde informatie, die een resource voor het uitvoeren van zijn taak nodig heeft, automatisch aan die resource kunnen doen toekomen.

4.4. TOETSING AAN FUNCTIONELE EISEN *WFM* SYSTEMEN

Het idee dat de bedrijfsprocessen zijn af te leiden uit de informatie die nodig is, en gebruikt wordt, tijdens het bedrijfsproces, is niet nieuw. Het is onder meer het uitgangspunt geweest van methoden als NIAM en FCO-IM. Of dit een goed uitgangspunt is bij het beschrijven en modelleren van organisaties is een vraag waarvan de beantwoording buiten dit onderzoek valt.

Een aantal bestaande *wfm* systemen is kort beschreven, en de conclusie is dat de (beschreven) functionaliteit daarvan uitgebreider is dan die van de ActieManager. Dit is geen onverwachte conclusie, aangezien de ActieManager het resultaat is van een afstudeerproject, en de beschreven systemen commercieel verkrijgbaar zijn. Wel kunnen we stellen dat de basisfunctionaliteit van een *wfm* systeem door de ActieManager aangeboden wordt; de punten waarop de ActieManager onderdoet zijn aanvullingen die ook door de rest van de systemen niet allemaal aangeboden worden.

Daarentegen blijkt dat in functionele zin de ActieManager prima voldoet aan de eisen die de Workflow Management Coalition stelt, uitgezonderd de samenwerking met andere informatiesystemen. Ook de aspecten die, naar verwachting, in de toekomst een belangrijke rol zullen gaan spelen, zijn vrijwel allemaal ondervangen; niet door de ActieManager zoals hier beschreven, maar wel door de DEMO-theorie. Dit betekent dat een *wfm* systeem op basis van de DEMO-theorie in ieder geval in functionele zin toekomst zou kunnen hebben.

HOOFDSTUK 4. FUNCTIONEEL ONTWERP VAN EEN *WFM* SYSTEEM

Hoofdstuk 5

Prototype: technisch ontwerp en implementatie

5.1 Inleiding

In hoofdstuk 4 is een functioneel ontwerp besproken van een *wfm* systeem op basis van de DEMO-methodiek. Dit functionele ontwerp is generiek, een beschrijving van de functionaliteit die een *wfm* systeem moet aanbieden. Het functionele ontwerp beschrijft een *wfm* systeem in termen van DEMO-concepten, en is volledig afgeleid uit de DEMO-theorie. Het ontwerp is getoetst aan de opgestelde criteria ten aanzien van *wfm* systemen, en blijkt goed mee te kunnen. Ook de vergelijking met commerciële systemen kan het doorstaan.

We hebben dus een generiek functioneel ontwerp van een *wfm* systeem, dat werkt op basis van een DEMO-model van een organisatie. Om de onderzoeksvraag te beantwoorden moeten we aantonen dat dit functionele ontwerp ook daadwerkelijk valt te bouwen, dat de beschreven functionaliteit daadwerkelijk door een informatiesysteem is aan te bieden. Dit hoofdstuk beschrijft een mogelijke uitwerking van het functionele ontwerp. Deze uitwerking bestaat uit een technisch ontwerp en een prototype implementatie. We beschrijven eerst het technische ontwerp, van zowel de software als de informatiestructuur, van een informatiesysteem dat de ontworpen functionaliteit aanbiedt. Daarna beschrijven we het prototype dat is gebouwd volgens dit technische ontwerp.

Dit hoofdstuk toont aan dat het opgestelde functionele ontwerp te bouwen is, en beantwoordt daarmee de derde deelvraag van de onderzoeksvraag. Het ontwerp is op globaal niveau opgesteld, en wellicht niet optimaal voor het ontwikkelen van een commercieel product, maar toont in ieder geval aan dat een implementatie van het functioneel ontwerp mogelijk is, en geeft daarvoor een oplossingsrichting.

Technisch ontwerp Een functioneel ontwerp beschrijft de functionaliteit die een informatiesysteem moet aanbieden aan zijn omgeving, aan zijn gebruikers. Een technisch ontwerp beschrijft hoe het informatiesysteem intern geconstrueerd is, zodanig dat het de beschreven functionaliteit aan zijn omgeving kan aanbieden. Een informatiesysteem bestaat uit software, hardware, data, mensen en procedures. We zullen ons richten op twee aspecten van een informatiesysteem die in het ontwerp de belangrijkste rol spelen: software en data. Uiteraard zijn de andere aspecten zeer relevant, maar vooral bij de

daadwerkelijke invoering en het dagelijkse gebruik van het informatiesysteem in een organisatie. Dit is in dit onderzoek niet aan de orde.

Het technisch ontwerp van het informatiesysteem bestaat in dit onderzoek dus uit de structuur van ten eerste de data die bewaard, bewerkt en ontsloten wordt, en ten tweede de software die dit verzorgt. Het functionele ontwerp is opgesteld met behulp van de DEMO-methodiek, hetgeen een zekere mate van houvast biedt bij het opstellen van deze twee ontwerpstructuren:

structuur van de software: het informatiesysteem is gemodelleerd als een geaggregeerde actor, waarvan de innerlijke werking onbekend is; slechts bekend is welke functionaliteit deze actor aanbiedt. Het technisch ontwerp van de software bestaat uit het uitwerken van deze geaggregeerde actor door de verschillende verantwoordelijkheden binnen deze geaggregeerde actor te onderkennen. Door verantwoordelijkheden toe te wijzen aan stukken software en de samenwerking tussen deze stukken te beschrijven, kunnen we een structuur aanbrengen in de software.

Dit ontwerp kan op verschillende abstractieniveaus gemaakt worden. Door het ontwerp op een hoog abstractieniveau te plaatsen, blijven er meer beslissingen over die tijdens de implementatie genomen moeten worden. Door het ontwerp op een laag abstractieniveau te plaatsen, moeten beslissingen soms van tevoren genomen worden zonder dat de daarvoor benodigde informatie en kennis aanwezig is. In dit onderzoek is gekozen voor een vrij hoog abstractieniveau; dit is geen principiële keuze, maar volgt uit het doel van het onderzoek: de opzet was niet om een optimaal *wfm* systeem (op basis van DEMO) te bouwen, maar om aan te tonen dat het bouwen van een dergelijk *wfm* systeem mogelijk is.

structuur van de informatie: de functionaliteit van een informatiesysteem bestaat uit het onthouden, verwerken en aanbieden van allerlei informatie. Het bepalen van de structuur van deze informatie is het tweede deel van het technisch ontwerp. Het functioneel ontwerp in dit onderzoek is afgeleid uit de beschrijving van een bedrijfssysteem (zie hoofdstuk 3): een werknemer die zijn taken bepaalt en uitvoert. Het informatiesysteem dat we ontwerpen ondersteunt dit bedrijfssysteem. Dit doet het door ten eerste een aantal actorrollen uit het bedrijfssysteem te vervullen informatie, en ten tweede informatie uit het bedrijfssysteem aan te bieden en te bewaren.

Voor het bepalen van de structuur van de informatie kunnen we de beschrijving van het bedrijfssysteem gebruiken. Onderdeel van het DEMO-model van het bedrijfssysteem is het feitenmodel, dat beschrijft welke transactieresultaten er zijn en welk verband deze onderling hebben. Dit feitenmodel is een basis voor de informatiestructuur van het informatiesysteem, aangezien het juist de informatie uit het bedrijfssysteem is die het informatiesysteem moet bewaren, verwerken en aanbieden.

We zullen daarom het technisch ontwerp als volgt beschrijven: eerst beschrijven we de structuur van de software, afgeleid van het functionele ontwerp. Hiervoor modelleren we eerst de samenwerking en constructie van de verschillende subsystemen in een DEMO-coördinatiemodel. Daarna modelleren we een gedeelte van de software in de Unified Modeling Language (UML), door middel van klassediagrammen. Vervolgens beschrijven we de structuur van de informatie, afgeleid uit feitenmodel van het bedrijfssysteem. Deze structuur modelleren we eerst op een abstract niveau, in een

ORM-model, en zetten we daarna om in een ERD-schema, dat de basis vormt voor het database-ontwerp.

Bij de ontwikkeling van het prototype is vanuit de opdrachtgever een aantal randvoorwaarden gesteld: ten eerste moet het prototype samenwerken met de Designer, een softwarepakket van ModulOr Technology waarmee DEMO-modellen zijn op te stellen: het prototype gebruikt de Designer als modelleergereedschap. Ten tweede is het prototype als een internet-applicatie geïmplementeerd, hetgeen betekent dat de gebruikers door middel van een internet-browser communiceren met het systeem. Hierdoor is er geen software-installatie nodig bij de gebruikers; een internet-browser is op alle kantoorcomputers standaard aanwezig.

Prototype implementatie Van het technisch ontwerp is een prototype implementatie gebouwd. Net als het technisch ontwerp is ook het prototype bedoeld om aan te tonen dat er een implementatie mogelijk is van het functioneel ontwerp. Het prototype is daarnaast voor de opdrachtgever een basis voor verdere productontwikkeling, om te komen tot een volwaardig *wfm* systeem op basis van DEMO.

In samenspraak met de opdrachtgever is ervoor gekozen om het prototype te implementeren met behulp van het .NET framework, in combinatie met een SQLServer database. De gebruikersinterfaces zijn gemaakt met ASP.NET, een manier om dynamische internetpagina's te programmeren; de logica van de software is geprogrammeerd in C#, een jonge object-georiënteerde programmeertaal die deel uitmaakt van het .NET framework. Voor communicatie met de database is gebruik gemaakt van ADO.NET, een manier om vanuit de software op een transparante manier met databases te communiceren.

Het prototype wordt na het technisch ontwerp beschreven, waarbij we aan zullen geven welke problemen bij de implementatie zijn bovengekomen, en welke elementen niet of niet volledig zijn uitgewerkt. Op basis hiervan stellen we aanbevelingen op voor de verdere ontwikkeling tot commercieel product.

5.2 Softwarestructuur

Het functionele ontwerp uit hoofdstuk 4 beschrijft de gewenste functionaliteit van het informatiesysteem. Dit informatiesysteem is gemodelleerd als een geaggregeerde actor die executor is van een aantal transacties. Deze transacties beschrijven de door het informatiesysteem aangeboden functionaliteit. We kunnen het informatiesysteem direct splitsen in elementaire actorrollen die ieder executor zijn van één enkele transactie. Dan wijzen we de verantwoordelijkheid voor een elementair stuk van de functionaliteit toe aan een bepaald stuk software. Dit is echter niet overzichtelijk bij het verder uitwerken van de software.

Meer voor de hand liggend is het gebruik van subsystemen, waarbij door middel van hiërarchische decompositie stukken van de functionaliteit die met elkaar te maken hebben, door één subsysteem worden aangeboden. Een subsysteem is verantwoordelijk voor een afgebakend stuk van de functionaliteit van het informatiesysteem. Subsystemen bieden deze functionaliteit aan de gebruikers aan of aan de rest van het informatiesysteem. Subsystemen zijn zelf geaggregeerde actorrollen, en zijn dus een verzameling van elementaire actorrollen.

5.2.1 Decompositie in subsystemen

Decompositie van een informatiesysteem betekent dat we functionaliteit zodanig over een aantal subsystemen verdelen dat de afhankelijkheid tussen die systemen minimaal is (een lage *coupling*) en de samenwerking tussen de onderdelen binnen elk systeem maximaal is (een hoge *cohesie*). Deze verdeling kan op verschillende gronden plaatsvinden. Als bijvoorbeeld delen van het systeem op verschillende locaties of verschillende platformen moet werken, dan kan dat de grond zijn voor het indelen in verschillende subsystemen. In dit systeem is geprobeerd een scheiding aan te brengen tussen de gebruikersinterface, de proceslogica en het databeheer. Binnen deze niveaus is de verantwoordelijkheid vervolgens in subsystemen opgedeeld:

- de verdeling is op het niveau van de gebruikersinterface gedaan op basis van de gebruikers en de door hun gevraagde functionaliteit.
- op het niveau van de proceslogica is de verdeling gemaakt op basis van het groeperen van de data waar de aangeboden functionaliteit betrekking op heeft.
- op het niveau van het databeheer is ervoor gekozen om één subsysteem te ontwerpen omdat databeheer in dit geval generiek is, en het dus niet raadzaam is om het te verdelen: het beheer van een gedeelte van de data is niet anders dan het beheer van de volledige data.

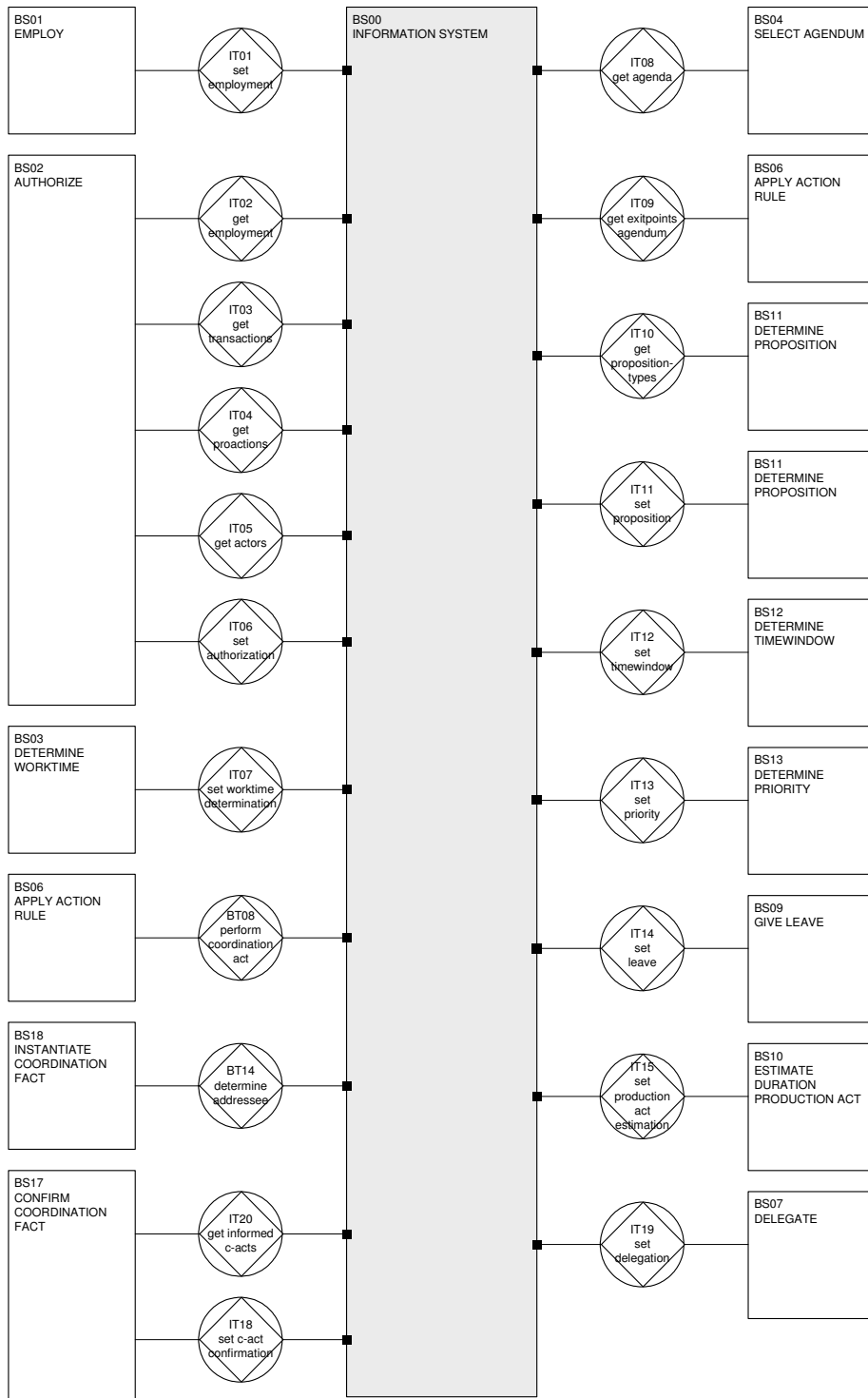
Per niveau zullen we de gemaakte verdeling in subsystemen bespreken en de verantwoordelijkheden van elk subsysteem. Daarna tonen we de samenwerking tussen de verschillende subsystemen.

Gebruikersinterface Op het niveau van de gebruikersinterface is de opdeling in subsystemen gebaseerd op de gebruiker: de stukken functionaliteit die een gebruiker over het algemeen met elkaar in verband brengt en tegelijkertijd gebruikt, zijn gebundeld. De term *substeem* zegt niets over de grootte of complexiteit van het element, juist de gebruikersinterfaces zijn over het algemeen niet erg complex van aard. De reden dat ze als apart subsysteem zijn gemodelleerd ligt in de totaal verschillende context waarin de gebruiker ze plaatst.

De totale functionaliteit van het totale informatiesysteem (uit hoofdstuk 4) is in figuur 5.1 ter herinnering afgebeeld. We zien hierin alle actorrollen die het informatiesysteem gebruiken en de functionaliteit die zij van het systeem verwachten. Deze actorrollen worden in de werkelijkheid door personen in de organisatie gespeeld. In het algemeen zal een aantal samenhangende actorrollen door dezelfde persoon, of hetzelfde type persoon, gespeeld worden.

We kunnen de gebruikers van het systeem dus indelen in een aantal categorieën, een aantal typen gebruikers, die bestaan uit een deelverzameling van de actorrollen uit figuur 5.1. We onderscheiden, op basis van de functionaliteit die ze van het informatiesysteem nodig hebben, vijf typen gebruikers van het informatiesysteem. In tabel 5.1 zijn vervolgens de actorrollen opgesomd die deel uitmaken van elk type gebruiker:

- de medewerker, die zijn taken bepaalt en uitvoert. Onder deze type gebruiker vallen bijvoorbeeld de actorrol *select agendum*, *determine proposition* en *apply action rule*.
- de personeelsmanager, die mensen in dienst neemt en autoriseert voor één of meer actorrollen. Onder deze type gebruiker vallen bijvoorbeeld de actorrollen *employ* en *authorize*.



Figuur 5.1: Functioneel ontwerp informatiesysteem, kopie van figuur 4.1

HOOFDSTUK 5. PROTOTYPE: TECHNISCH ONTWERP EN IMPLEMENTATIE

- de prioriteitsmanager, die bepaalt hoe hoog de prioriteit is van binnenkomende verzoeken en hoe snel deze dus afgehandeld zouden moeten worden. Dit is een taak die over het algemeen niet door (algemene) medewerkers zelf wordt afgehandeld, aangezien dit een tactische beslissing is. Deze beslissing vereist informatie waarover zij niet beschikken, zoals over de andere verzoeken die binnen zijn gekomen, de tactische doelen van de organisatie, beschikbaarheid van medewerkers en de financiële positie van de organisatie.
- de werktijdmanager, die bepaalt wie wanneer verlof heeft (bijvoorbeeld wegens ziekte of vakantie) en hoeveel tijd het uitvoeren van een productieve acties kost. Ook dit is een tactische functie, die niet door de algemene medewerkers zelf wordt uitgevoerd, maar door een daarin gespecialiseerd persoon. Het valt buiten de dagelijkse werkzaamheden van een algemene medewerker.
- de systeembeheerder, die het systeem onderhoudt en beheert. Deze staat in het functionaliteitsoverzicht niet genoemd, omdat zijn taak *over* het systeem gaat, en niet deel uitmaakt *van* het systeem. Het had evenwel wel opgenomen kunnen worden. De systeembeheerder kent bijvoorbeeld gebruikers toe aan het systeem; dit zou in de toekomst gekoppeld kunnen worden aan de personeelsadministratie, hetgeen technisch niet ingewikkeld is. De systeembeheerder heeft daarnaast de mogelijkheid het DEMO-model van de organisatie te importeren, en daarmee het *wfm* systeem geschikt maken voor een specifieke organisatie. Dit DEMO-model is afkomstig van de Designer, het softwarepakket van ModulOr Technology waarmee DEMO-modellen zijn te maken. Indien er in de organisatie iets wezenlijks verandert, kan het DEMO-model in de Designer aangepast worden en opnieuw in de ActieManager ingelezen worden.

<i>type gebruiker</i>	<i>bevat actorrol</i>
medewerker	: BS04 select agendum
	: BS06 apply action rule
	: BS07 delegate
	: BS11 determine proposition
	: BS12 determine timewindow
	: BS17 confirm coordination act
	: BS18 instantiate coordination fact
personeelsmanager	: BS01 employ
	: BS02 authorize
	: BS03 determine worktime
prioriteitsmanager	: BS13 determine priority
tijdmanager	: BS09 give leave
	: BS10 estimate duration production act

Tabel 5.1: Type gebruikers en de actorrollen daarin

Gebaseerd op het type gebruiker komen we op het niveau van de gebruikersinterfacés tot een verdeling in vijf subsystemen, vijf gebruikersinterfacés: een *user interface* (UI) voor de personeelsmanager, een UI voor de prioriteitsmanager, een UI voor de werktijdmanager, een UI voor de systeembeheerder, en tenslotte een UI voor de werknemer.

Proceslogica Op het niveau van de proceslogica is de taak van de software het implementeren van bedrijfsregels en procedures, zodat de informatie op de juiste manier verwerkt wordt. De proceslogica is de plek waar ‘het echte werk gebeurt’. De gebruikersinterface bewerkt de informatie zodat het leesbaar en handelbaar is door de gebruikers. De data laag bewerkt de informatie zodat het uniform opgeslagen en bewaard kan worden op een persistent medium. Op het niveau van de proceslogica wordt de data uit de data laag verwerkt en bewerkt volgens de bedrijfsregels, procedures en algoritmen.

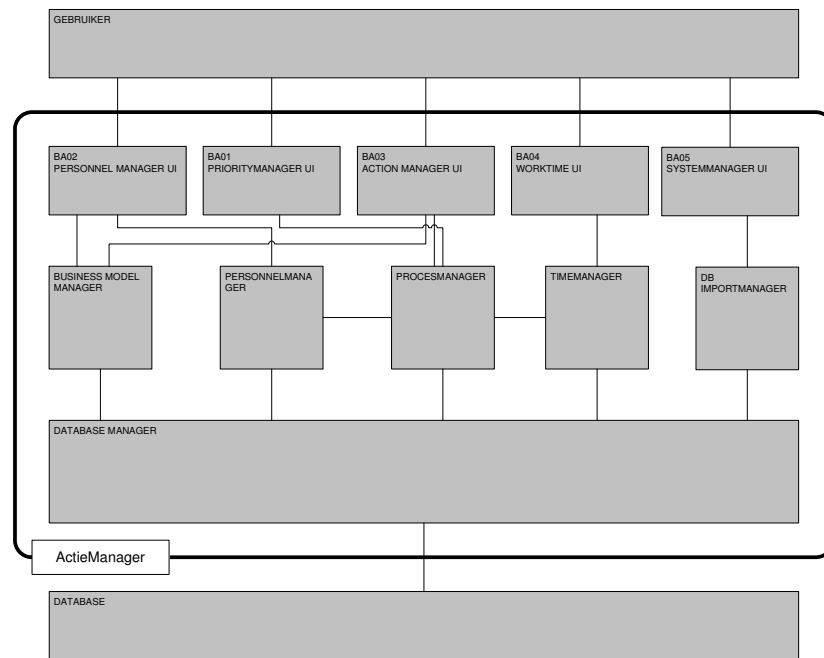
Op het niveau van de proceslogica (of *business rules*) is de opdeling in subsystemen gebaseerd op de informatie die door de verschillende elementen verwerkt wordt. Deze subsystemen implementeren de proceslogica van de organisatie, en bewerken en beheren daarvoor bepaalde informatie. Door deze informatie te groeperen komen we tot de volgende vier verantwoordelijkheden:

- beheer van het bedrijfsmodel: het beheren van alle informatie over het DEMO-model van de organisatie die ondersteund wordt. Dit bestaat onder andere uit het bewaren en vrijgeven van informatie over de actorrollen en transacties die in de organisatie gemodelleerd zijn.
- beheer van de personeelsgegevens: het beheren van alle informatie over de in de organisatie werkzame personen, hun bevoegdheden en werktijden. Hieronder valt ook het beheer van de toegang tot het systeem; niet alle gebruikers hebben immers toegang tot alle onderdelen van het systeem. De bevoegdheden die aan een gebruiker worden toegekend zijn afhankelijk van de rol die deze gebruiker heeft in het systeem.
- beheer van de tijdsgegevens: het beheren van alle informatie over de beschikbaarheid van medewerkers, de tijdsduur van productieve acties. Hieronder valt bijvoorbeeld ook het bepalen van de medewerker die de grootste beschikbaarheid heeft, of de medewerker wiens beschikbaarheid dichtbij een bepaald tijdsvenster ligt.
- beheer van het bedrijfsproces: dit is de workflow engine, verantwoordelijk voor het beheren van het bedrijfsproces. Hieronder valt het bepalen van de adressee van een coördinatieve actie, het bepalen van de agenda van een medewerker en het beheren van de bevestigingen die gedaan zijn.

Databeheer Het databeheer (het bewaren en beschikbaar maken van persistente data) valt in dit ontwerp onder de verantwoordelijkheid van een apart subsysteem. Het schakelt hiervoor een database management systeem in; de taak van het databeheer subsysteem bestaat daardoor uit twee taken: ten eerste zorgt het ervoor dat de data-opslag voor de rest van het systeem transparant is, onafhankelijk van het gebruikte database management systeem.

De tweede taak komt voort uit een vaak voorkomend paradigmaverschil tussen software en databeheer: software wordt object-georiënteerd ontworpen, de database relationeel-georiënteerd. Het object-georiënteerde paradigma is een manier om complexiteit in de software de baas te worden door verantwoordelijkheden, data en functionaliteit te groeperen. Er bestaan tegenwoordig wel object-georiënteerde databases, maar deze hebben vaak last van prestatieproblemen en zijn (nog) niet wijdverspreid commercieel verkrijgbaar. De meeste databases zijn relationeel, en zijn opgebouwd uit tabellen van data en relaties daartussen.

Het groeperen van verantwoordelijkheden, functionaliteit en data in objecten leidt in het algemeen tot een andere verdeling dan het groeperen van de data in relationele tabellen. Dit betekent dat er een transformatie moet plaatsvinden om persistente software objecten in een relationele database op te slaan. Dit is de tweede taak van ons databeheer subsysteem.



Figuur 5.2: Decompositie in subsystemen

Subsystemen In figuur 5.2 zijn de verschillende subsystemen gemodelleerd¹ en de samenwerking die er tussen deze subsystemen bestaat. De grafische interfaces communiceren met de gebruikers en werken samen met de subsystemen voor proceslogica, die de gebruikerswensen kunnen uitvoeren:

- de *personeelsmanager UI*, waarmee de personeelsmanager aan het systeem kan doorgeven wie er in dienst is gekomen en welke actorrollen hij gaat vervullen, werkt samen met het subsysteem *business model manager*, om informatie te krijgen over de transacties en actoren in het systeem, en het subsysteem *personeelsmanager*, om informatie door te geven over de personen die in dienst zijn getreden, de actorrollen die zij vervullen, en de hoeveelheid tijd die zij daarvoor beschikbaar hebben.
- de *prioriteitsmanager UI*, waarmee de prioriteitsmanager aan het systeem kan doorgeven wat de prioriteit is van een bepaald verzoek, werkt samen met de procesmanager. De procesmanager is immers verantwoordelijk voor het beheer van

¹vanwege de beperkte beschikbare tijd hebben we de samenwerking in subsystemen niet getoond in DEMO-transacties. Deze transacties zijn in de hieropvolgende uitwerking van de subsystemen wel gemodelleerd. Deze transacties kunnen door de lezer in figuur 5.2 ingevuld worden.

het bedrijfsproces, waaronder de coördinatieve feiten en de prioriteit daarvan, vallen.

- de *systeembeheerder UI*, waarmee de systeembeheerder dynamische data kan verwijderen uit het systeem en een nieuw of veranderd DEMO-model kan importeren, werkt samen met de importmanager.
- de *werktijdmanager UI*, waarmee de tijdmanager aan het systeem kan doorgeven wanneer personen beschikbaar zijn, werkt samen met het subsysteem *tijdsbeheer*, die immers verantwoordelijk is voor het beheren van alle tijdsgegevens.
- de *werknemer UI*, waarmee werknemers hun agenda kunnen raadplegen, erop kunnen reageren door te communiceren met andere werknemers, werkt samen met twee subsystemen. Ten eerste werkt het samen met het subsysteem *procesbeheer*, dat verantwoordelijk is voor het beheren van het bedrijfsproces, waaronder ook de agenda van alle personen valt, en hun reacties daarop. Ten tweede werkt het samen met het subsysteem *business model manager*, om aan de gebruikers te kunnen doorgeven welke transacties zij kunnen initiëren tijdens het afhandelen van een agendum, welke propositietype hoort bij een bepaalde transactie, oftewel, welke (re)actiemogelijkheden zij volgens het organisatiemodel hebben.

De subsystemen voor de proceslogica communiceren met het databasesubstelsysteem, waarbij ze data opvragen en laten opslaan. Het databasesubstelsysteem communiceert hiertoe met het database management systeem (in dit geval is dat een SQLServer) die de daadwerkelijke opslag van de data op zich neemt.

5.2.2 Verantwoordelijkheden binnen subsystemen

De verschillende subsystemen en de samenwerking daartussen zijn nu bepaald. De subsystemen zijn intern ook in verantwoordelijkheden te groeperen, hetgeen we in een aantal DEMO-diagrammen zullen tonen. Deze verantwoordelijkheden vormen de basis voor het UML-ontwerp van de klassen binnen de subsystemen. Klassen zijn in het object-georiënteerde paradigma immers brokken software, verantwoordelijk voor het aanbieden van een bepaalde hoeveelheid functionaliteit en het daaruit volgende beheer van data.

De functionaliteit van het informatiesysteem, die in figuur 5.1 beschreven staat, is verdeeld over een aantal subsystemen. In figuur 5.2 zijn de gebruikersinterfaces beschreven waarmee deze functionaliteit wordt aangeboden. Voor het daadwerkelijk uitvoeren van deze functionaliteit zijn actoren binnen de subsystemen op het niveau van de proceslogica verantwoordelijk. Dit is getoond in een DEMO-model, dat voor het gemak in drie afzonderlijke diagrammen is weergegeven: in figuur 5.3 zien we de uitwerking van de subsystemen *BUSINESSMODELMANAGER* en *PERSONNELMANAGER*; in figuur 5.4 de uitwerking van de *PROCESMANAGER* en de *TIMEMANAGER*; en in figuur 5.5 tenslotte de *DATABASEMANAGER*.

Alle actorrollen uit het functioneel ontwerp gaan transacties aan met de gebruikersinterfaces, om gebruik te maken van de aangeboden functionaliteit. In figuur 5.4 is voor het gemak een systeemactor gemodelleerd, *Worker*, die de verzameling is van de erin geschreven actorrollen. Alle transacties uit het functionele ontwerp worden aangeboden door een bepaalde gebruikersinterface, zodat het totale informatiesysteem de gehele beschreven functionaliteit aanbiedt.

HOOFDSTUK 5. PROTOTYPE: TECHNISCH ONTWERP EN IMPLEMENTATIE

Zoals in figuur 5.5 is weergegeven, initiëren alle actoren die data willen opslaan en opvragen een transactie met de DATABASEMANAGER. In de diagrammen zijn deze transacties om redenen layout niet getoond.

BusinessModelManager De PERSONNELMANAGERUI biedt ten eerste de mogelijkheid aan de gebruikers om te informeren naar de transacties, proacties en actoren in het bedrijfsmodel. Aangezien de PERSONNELMANAGERUI dat zelf niet weet, gaat het hiervoor een aantal transacties aan met de BUSINESSMODELMANAGER, die zoals gezegd verantwoordelijk is voor het beheer van het bedrijfsmodel. De BUSINESSMODELMANAGER biedt daarom transacties om de actoren, transacties en proacties in het bedrijfsmodel op te vragen.

Daarnaast is de BUSINESSMODELMANAGER executor is van nog twee transacties, de initiator daarvan is de ACTIONMANAGERUI, zoals we in figuur 5.4 kunnen zien. Deze heeft informatie nodig over de exit-points van de verschillende actieregels, dus welke reacties op een agendum er volgens het bedrijfsmodel mogelijk zijn. Daarnaast heeft deze informatie nodig over de mogelijke propositietypen, dus welke propositietype deel uitmaakt van een transactie. Met deze informatie kan de ACTIONMANAGERUI de gebruiker ten eerste een reactie laten selecteren uit de mogelijkheden die het uit het bedrijfsmodel gegeven worden, zodat de gebruiker bijvoorbeeld niet een belofte laat volgen op een verklaring. Daarnaast kan de ACTIONMANAGER hiermee bij het initiëren van een interactie aan de gebruiker doorgeven welk propositietype daarbij hoort, zodat de gebruiker deze alleen nog maar hoeft te instantiëren (een bepaalde waarde geven).

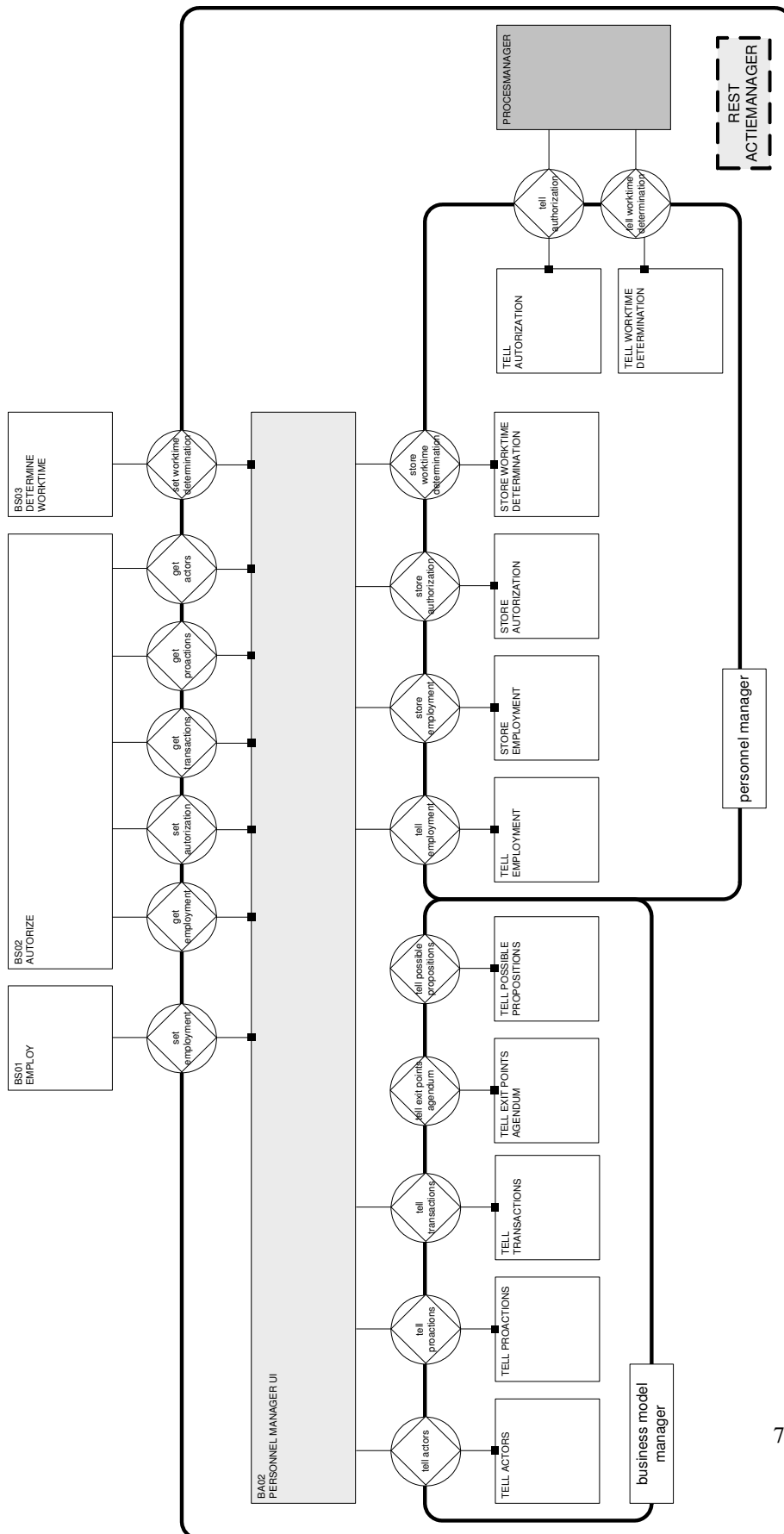
PersonnelManager De PERSONNELMANAGERUI maakt het mogelijk om aan te geven dat mensen in dienst zijn getreden, en geautoriseerd voor een aantal actorrollen. De PERSONNELMANAGER, die verantwoordelijk is voor het beheer van alle personeelsgegevens, biedt een aantal transacties aan om deze informatie te bewaren.

Daarnaast biedt de PERSONNELMANAGER een aantal transactie aan de PROCESMANAGER om deze informatie op te vragen. De PROCESMANAGER moet immers weten wie, voor welk tijdsdeel, voor welke actorrol geautoriseerd is, zodat hij kan bepalen van de geschikte persoon om een coördinatieve actie aan te richten, het *schedulen*.

TimeManager De WORKTIMEUI biedt de gebruiker de mogelijkheid om zijn aanwezigheid aan te geven. Daarnaast maakt de WorkTimeUI het mogelijk om van allerlei werkzaamheden (productie-acties) aan te geven wat de geschatte tijdsduur ervan is. Beiden worden gebruikt bij het bepalen van de adressee van coördinatieve acties. De TIMEMANAGER biedt daarom twee transacties aan voor het bewaren van deze informatie. Daarnaast biedt de TIMEMANAGER twee transacties aan de PROCESMANAGER voor het opvragen van deze informatie. Die heeft deze informatie nodig bij het *schedulen*.

ProcesManager De ACTIONMANAGERUI laat gebruikers hun agenda bekijken, erop reageren en delegeren aan andere gebruikers. Ook toont de ACTIONMANAGERUI de gebruikers welke coördinatieve acties zij moeten bevestigen, en biedt hen ook de mogelijkheid dat te doen.

De functionaliteit die hiervoor nodig is wordt door de PROCESMANAGER aangeboden, die eigenlijk de workflow engine van het informatiesysteem vormt. De PROCESMANAGER biedt transacties voor het opvragen van nog niet bevestigde, aan de gebruiker gerichte, coördinatieve acties, en voor het bevestigen van deze coördinatieve acties.



Figuur 5.3: De subsystemen BUSINESSMODELMANAGER en PERSONNELMANAGER

staat het type-niveau van de informatie beschreven, de ActieManager maakt daar instanties van. De database van de ActieManager wordt op type-niveau gegeven door het DEMO-model, en op instantie-niveau door de handelingen van de gebruikers. De DBIMPORTMANAGER zal aan de Designer database vragen om de data die het DEMO-model vormt van de organisatie, en deze vervolgens in de database van de ActieManager opslaan.

5.2.3 Ontwerpbeslissingen binnen subsystemen

De uitwerking van de verschillende verantwoordelijkheden binnen de subsystemen in klassen is beschreven in een aantal UML-diagrammen. Deze vormen niet het complete ontwerp van het informatiesysteem, maar beschrijven in het algemeen enkele ontwerpbeslissingen. Bij deze ontwerpkeuzes is uitgegaan van de beslissing om het geheel met behulp van het .NET framework te implementeren.

De UML-diagrammen vallen over het algemeen samen met de gegeven verdeling in subsystemen. Het zijn echter geen volledig uitgewerkte ontwerpen van alle klassen binnen de verschillende subsystemen, maar beschrijven op een wat hoger niveau de belangrijkste ontwerpbeslissingen.

Een aantal subsystemen, zowel op het niveau van de proceslogica als van het databeheer, maakt gebruik van zogenaamde *stored procedures*. Dit zijn stukken code die in de database zitten, en door het database management systeem worden uitgevoerd. Het zijn eigenlijk stukken software die, bijvoorbeeld uit efficiëntie-overwegingen, in de database ingebed zijn. Dit betekent dat een deel van de verantwoordelijkheid van deze subsystemen niet door software-klassen wordt geïmplementeerd, maar door deze stored procedures. Voor de volledigheid zijn alle gebruikte stored procedures in appendix B gedefinieerd.

Gebruikersinterfaces De gebruikersinterfaces worden met behulp van ASP.NET geïmplementeerd, waarbij elke gebruikersinterface door een aparte *.aspx* pagina wordt voorgesteld. ASP.NET is een manier om dynamische HTML-pagina's te maken, dit betekent dat elke gebruiker een op maat gemaakte pagina te zien krijgt. In figuur 5.8 staat beschreven hoe de verschillende gebruikersinterfaces samenwerken met de rest van het systeem. Daarin zijn een aantal elementen te onderscheiden.

- *Inloggen*: het is noodzakelijk dat het informatiesysteem de identiteit van de gebruiker kent, het systeem is immers op individuele personen afgestemd. Het is daarom nodig dat elke gebruiker zich op de één of andere manier aan het systeem identificeert. Hiertoe is gebruik gemaakt van een standaard oplossing uit het .NET framework, genaamd ASP.NET WebForms. Dit houdt in dat elke keer dat een gebruiker een pagina (een gebruikersinterface) opvraagt terwijl hij nog niet ingelogd is, hij wordt doorgestuurd naar de *LoginUI*.

Als het informatiesysteem wordt ingezet als intranet-toepassing en er sprake is van een bedrijfsnetwerk waarop alle gebruikers zich reeds geïdentificeerd hebben, dan kan van die informatie gebruik worden gemaakt. Als de gebruikers bijvoorbeeld altijd inloggen via een *Active Directory* server, dan hoeven ze niet meer handmatig in dit informatiesysteem in te loggen, maar kunnen die gegevens ook voor dit informatiesysteem worden gebruikt. De inlog-procedure is gemakkelijk aan de situatie te passen, in dit prototype is dus gekozen voor het handmatig inloggen; daarbij is wel de mogelijkheid gegeven de inloggegevens op de lokale computer te bewaren, zoals we later zullen zien.

- *ActionManagerUI*: dit is voor het dagelijkse werk de belangrijkste gebruikersinterface. Deze vraagt aan de procesmanager de agenda van de gebruiker, de mogelijke reacties die de gebruiker kan uitoefenen op een bepaald agendum en de mogelijkheden van deze gebruiker om zelfstandig interacties te starten. Of een gebruiker een interactie mag initiëren is afhankelijk van de door hem gespeelde actorrollen en hun bevoegdheden volgens het DEMO-model van de organisatie. Als een gebruiker een interactie wil initiëren dan roept de *ActionManagerUI* de hulp in van een andere gebruikersinterface, de *PFactUI*.
- *PFactUI*: deze gebruikersinterface moet ervoor zorgen dat alle gegevens die voor de initiatie van een interactie noodzakelijk zijn, door de gebruiker ingevuld worden. Deze gegevens bestaan uit de gewenste executor, tijdsvenster en propositie. De initiator van de interactie is bekend, namelijk de ingelogde gebruiker. De executor van de transactie kan door de gebruiker worden geselecteerd uit een door het informatiesysteem opgestelde lijst. Het informatiesysteem selecteert deze personen op basis van de actorrollen die zij mogen spelen (ze moeten uiteraard bevoegd zijn om de gewenste interactie uit te voeren) en de tijd die zij beschikbaar hebben.

In dit prototype is ervoor gekozen om de gebruiker het laatste woord te geven bij het selecteren van de executor. De ActieManager maakt een (gesorteerde) lijst, maar de gebruiker kan daar altijd van afwijken. Dit kan uiteraard eenvoudig aangepast worden zodat de ActieManager niet meer een selectie aanbiedt, maar zelf een keuze maakt.

Het laten invullen van de gewenste propositie van de interactie lijkt op het eerste gezicht eenvoudig. Elke DEMO-interactie kent een propositietype, een productiefeytype dat in de interactie tot stand moet worden gebracht. Beschouw hiervoor het volgende:

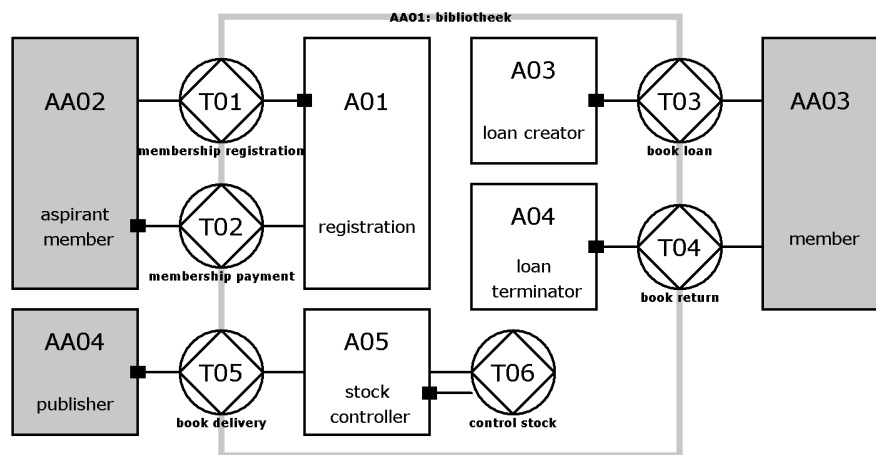
In figuur 5.6 is het coördinatiemodel van een bibliotheek gemodelleerd, het bijbehorende feitenmodel is in figuur 5.7 gemodelleerd. In deze bibliotheek kunnen aspirant-leden een lidmaatschap verkrijgen door een transactie (T01) aan te gaan met een actor in de bibliotheek. De propositie van deze transactie is '*er is een Lidmaatschap L*', zoals we in het feitenmodel kunnen zien. Bij het initiëren van deze transactie moet de propositie geïnstantieerd worden, en luidt de propositie bijvoorbeeld '*er is een Lidmaatschap #53446*'.

Maar als in de ActieManager een aspirant-lid deze transactie zou willen initiëren dan is het invullen van een lidmaatschapsnummer niet genoeg. In het feitenmodel is immers te zien dat er aan een *Lidmaatschap* een verplichte-rol regel verbonden is: elk lidmaatschap moet een rol spelen in de relatie '*Persoon ... is het lid van Lidmaatschap ...*'. Het aspirant-lid moet dus volgens het feitenmodel ook dat feitype invullen en aangeven over welke persoon het lidmaatschap gaat.

Het invullen van de propositie van een interactie kan dus een ingewikkelde zaak worden, waarbij allerlei verplichte-rollen ingevuld moeten worden. Dit is geheel afhankelijk van het feitenmodel, en de regels die daarin ten aanzien van de productiefeyten gesteld worden. De *PFactUI* is er dus voor verantwoordelijk dat de gebruiker de mogelijkheid krijgt om alle verplichte informatie in te vullen die voor het initiëren van een interactie nodig is.

Hiervoor maakt het gebruik van de klasse *RelationRepresentation*, de representatie van een feit. Een feit is in de DEMO-theorie een relatie tussen aantal objecten. Deze objecten komen zelf altijd tot stand komen als gevolg van een transactie, tenminste in het uitgangspunt van de DEMO-theorie. Omdat het opgestelde model niet de gehele wereld bevat, kennen we van sommige objecten de transacties waardoor ze tot leven komen wel, zoals ‘de contributie voor L is betaald’ of ‘B is geleverd’. Van andere objecten, externe objecten, kennen we deze transacties niet; deze objecten komen buiten de systeemgrens van het model tot stand, en bestaan voor ons gewoon.

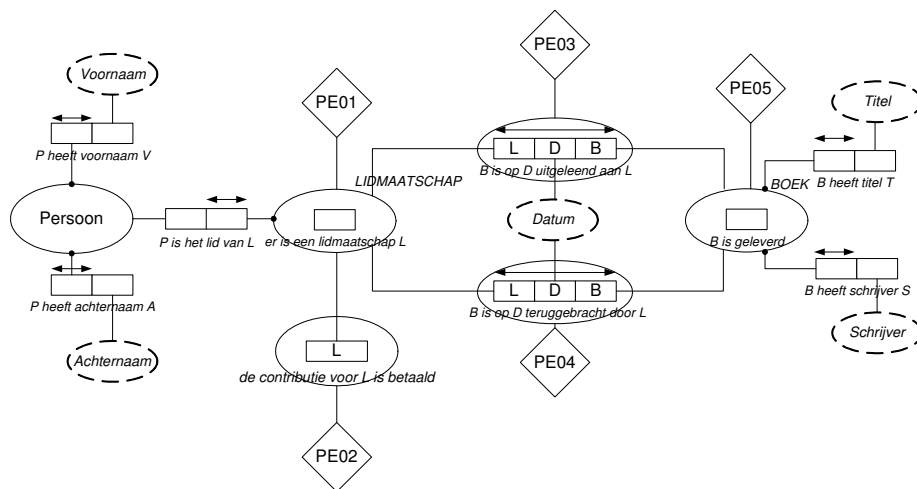
Voor het refereren naar objecten die tot stand komen als gevolg van een transactie maakt de *RelationRepresentation* gebruik van de klasse *ReferringRoleChoice*, een soort choice-box, waarmee gebruikers het object kunnen kiezen waarnaar ze willen refereren. De keuze voor de objecten wordt bepaald door het resultaat van de transacties die dat object creëren, alleen de objecten die in de propositie van transactie in de fase *geaccepteerd* zijn, bestaan, en kunnen worden gekozen.



Figuur 5.6: Voorbeeld coördinatie-model bibliotheek

- *ResourceManager*: in het ontwerp is rekening gehouden met een eventuele wens om het systeem in verschillende talen aan te bieden. Als een gebruikersinterface een tekst aan de gebruiker wil tonen, dan vraagt hij aan de *ResourceManager* om de juiste zinexpressie hiervan. De *ResourceManager* kent de gewenste taal van de gebruiker (zou bijvoorbeeld bij het inloggen ingesteld kunnen worden) en haalt uit een bestand met teksten de zinexpressie in de gewenste taal op.
- *PersonnelManagerUI*, *TimeManagerUI* en *SystemManagerUI*: de werking van deze gebruikersinterfaces is niet bijzonder. Hun verantwoordelijkheden en de daarvoor benodigde samenwerking met andere subsystemen zijn reeds beschreven.

BUSINESSMODELMANAGER De verantwoordelijkheid van de BUSINESSMODELMANAGER bestaat uit het beheren van het bedrijfsmodel. Het bedrijfsmodel is voor de



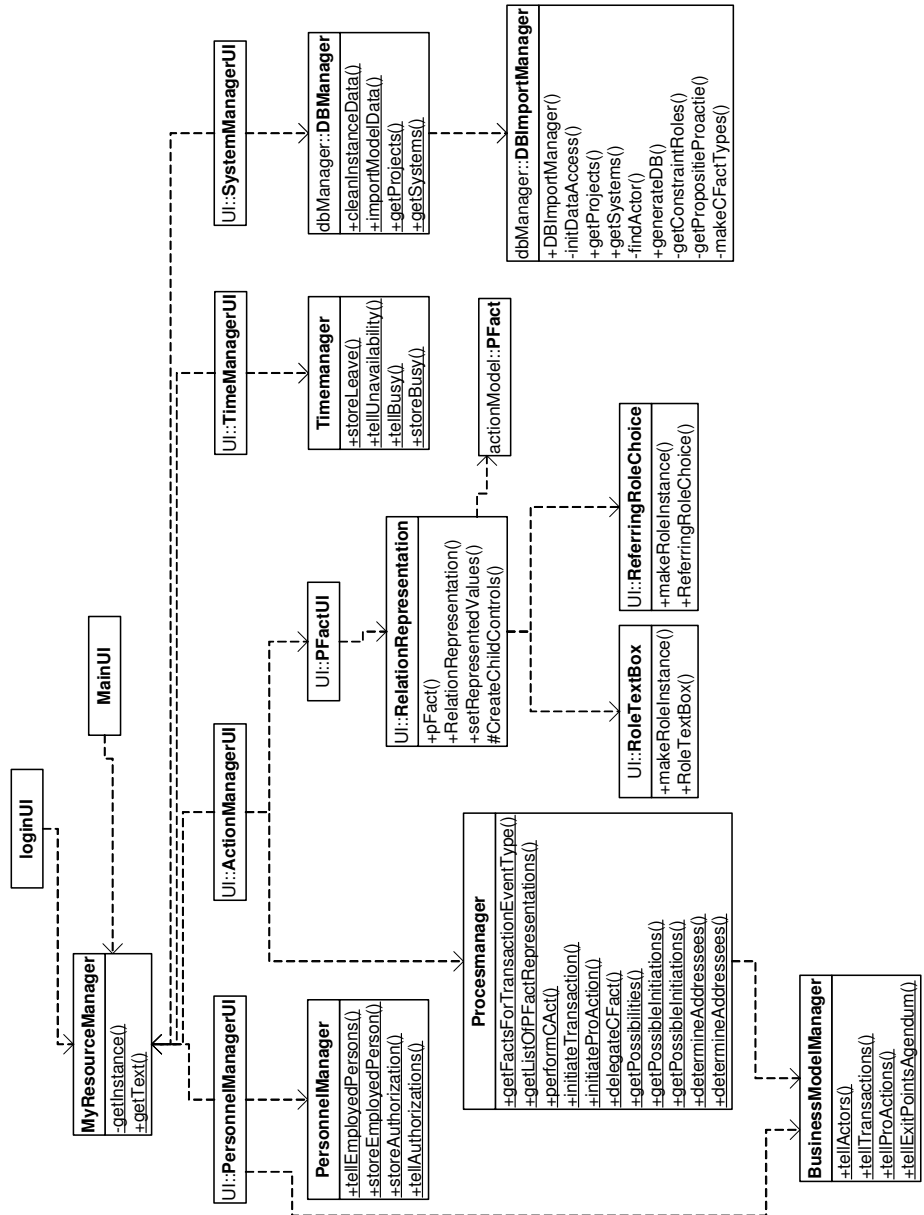
Figuur 5.7: Voorbeeld feitenmodel bibliotheek

ActieManager een informatiebron waaruit alleen gelezen wordt: het aanpassen of opstellen van het bedrijfsmodel vindt plaats in de Designer. Het bedrijfsmodel biedt een kader voor het hele systeem: het bepaalt onder andere de reactiemogelijkheden van de gebruikers. De klassen in de *BUSINESSMODELMANAGER*, waarvan het model in figuur 5.9 is getoond, zijn daarom vooral een representatie van data, en lijken sterk op de informatiestructuur (zie sectie 5.3) van het informatiesysteem. In elke klasse is een methode te zien 'writeToDB()', waarmee objecten zichzelf in de database kunnen laten schrijven. Meer hierover komt aan de orde bij de bespreking van de *DATABASEMANAGER*.

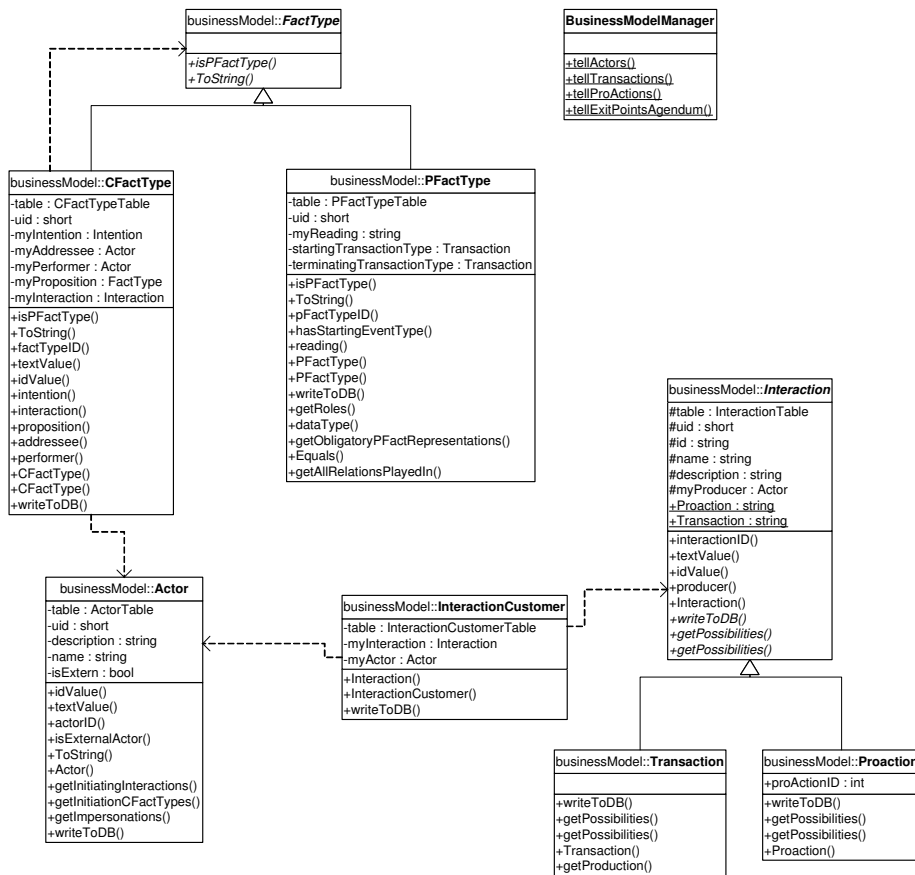
De klasse *BusinessModelManager* is een facade klasse, die aan andere klassen de functionaliteit van het gehele subsysteem aanbiedt, zoals in figuur 5.8 is te zien. Aan deze klasse kunnen de gebruikersinterfaces vragen welke actoren, transacties, en proacties er in het bedrijfsmodel zitten, en wat de exit-points van een agendum zijn. De methoden die deze klasse aanbiedt zijn de implementaties van de transacties (zie figuur 5.3) waarvan dit subsysteem de executor is. Het antwoord dat de *BusinessModelManager* geeft bestaat uit de objecten uit figuur 5.9, de objecten die het bedrijfsmodel representeren. Aan deze objecten kunnen vervolgens weer vragen worden gesteld; aan een transactie kan bijvoorbeeld gevraagd worden wie zijn executor is.

PROCESMANAGER en PERSONNELMANAGER Een aantal klassen uit het subsysteem *PROCESMANAGER* zijn in figuur 5.10 te zien. In dezelfde figuur zijn ook een aantal klassen uit het subsysteem *PERSONNELMANAGER* weergegeven. Ook voor deze subsystemen geldt weer dat er een facade klasse is ontworpen, respectievelijk de *ProcesManager* en de *PersonnelManager*, die de functionaliteit van deze subsystemen aan de buitenwereld aanbiedt.

DATABASEMANAGER Het subsysteem *DATABASEMANAGER* is verantwoordelijk voor twee dingen: ten eerste het op beheer van persistente data, op een zodanige manier dat de onderliggende database voor de rest van het systeem niet relevant is; ten tweede het verzorgen van een transformatie tussen de (object-georiënteerde) persistente klassen

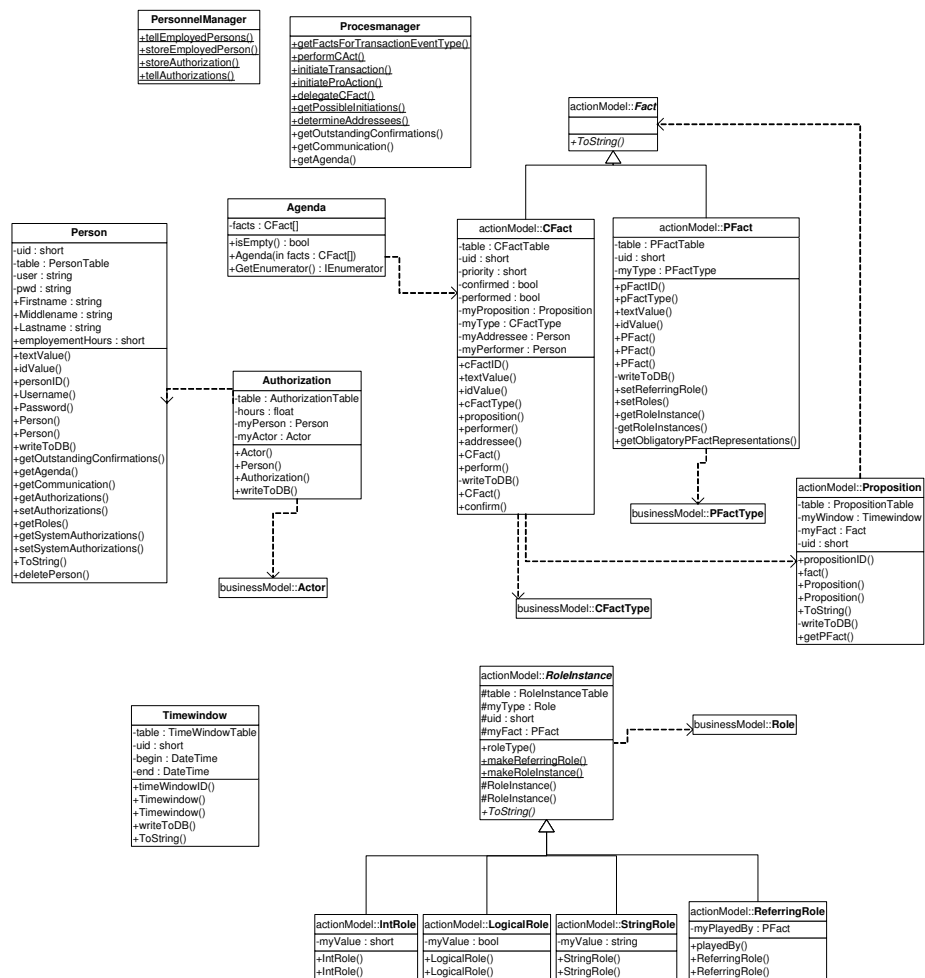


Figuur 5.8: Gebruikersinterfaces



Figuur 5.9: Klassen in de BUSINESSMODELMANAGER

HOOFDSTUK 5. PROTOTYPE: TECHNISCH ONTWERP EN IMPLEMENTATIE



Figuur 5.10: Klassen in de PROCESMANAGER en PERSONNELMANAGER

en de (relationele) database tabellen. Voor de invulling van deze verantwoordelijkheid van dit subsysteem is de keuze om dynamisch gegenereerde internetpagina's als gebruikersinterface te gebruiken, van belang:

Sommige internetpagina's worden statisch aangeboden, hetgeen betekent dat alle gebruikers dezelfde pagina te zien krijgen. Een bibliotheek kan bijvoorbeeld de catalogus online zetten, in één lange lijst, of wat iets handiger is, verdeeld over bijvoorbeeld 26 pagina's voor elke beginletter van de schrijver. Als de gebruiker met zijn internetbrowser naar de gewenste pagina surft verzoekt hij de internetserver om de gewenste html-pagina naar hem te versturen. De internetserver weet in principe niet of de gebruiker al vaker deze pagina verzocht heeft, of dit juist voor het eerst doet.

Dynamisch gegenereerde pagina's bestaan al heel lang (denk aan cgi-scripts) en zijn veel krachtiger, aangezien men de getoonde informatie aan de gebruiker kan aanpassen. De bibliotheek kan daarmee bijvoorbeeld een zoekveld aanbieden, en alleen dat gedeelte van de catalogus kunnen aanbieden dat met de zoektermen overeenkomt. De werking van dynamische internetpagina's is als volgt: het verzoek van de gebruiker om een bepaalde html-pagina komt wederom bij de internetserver binnen, maar nu met enkele parameters met daarin de door de gebruiker ingevulde waarde van het zoekveld (of andere ingevulde informatie). De internetserver stelt dan op grond van die parameters (en eventueel andere variabelen zoals de tijd) de html-pagina op, en verstuurt die naar de gebruiker. Ook hier weet de internetserver niet of de gebruiker deze pagina vaker heeft opgevraagd, of welke parameters de gebruiker ooit heeft ingevuld. De internetserver ziet slechts losse aanvragen voor pagina's, maar kent daartussen geen verband.

Dit laatste kan worden opgelost door het gebruik van *sessies*, hetgeen in ASP.NET automatisch gebeurt. Doordat de gebruiker bij elk verzoek ook een hem uniek identificerende sessievariabele meestuurt kan de server nu wel het verband tussen verschillende verzoeken van de gebruiker inzien. Het probleem is echter niet geheel verholpen: de software-objecten die bij het ene verzoek in het geheugen van de internetserver bekend zijn, zijn bij het tweede verzoek verdwenen. De verzoeken staan namelijk eigenlijk los van elkaar, ze zijn weliswaar met elkaar verbonden door een gemeenschappelijke sessievariabele, maar de internetserver wordt bij elk verzoek opnieuw geïnitieerd. De oplossing daarvoor is dat bij elk verzoek alle benodigde software-objecten opgebouwd worden uit de data in de database. Dit laatste valt onder de verantwoordelijkheid van de DATABASEMANAGER.

We gebruiken ADO.NET om met de database te communiceren. In ADO.NET wordt data (bijvoorbeeld de inhoud van één tabel, of het resultaat van een query over meerdere tabellen) gerepresenteerd door een zogenaamde *DataSet*. Een DataSet bestaat uit rijen met bepaalde velden. Een DataSet is ontkoppeld van zijn bron: de data wordt door een *DataAdapter*, die een SQL-query op de bron uitvoert, in de DataSet gezet. Daarna wordt de verbinding verbroken en kan er van alles met de DataSet gebeuren. Het is mogelijk om zelf subklassen van een DataSet te ontwerpen, die allerlei attributen en methoden bezitten, die is toegespitst op een bepaalde soort data (bijvoorbeeld uit één tabel). Op die manier kan tijdens het compileren worden bepaald of er geen verkeerde type data in of uit de DataSet wordt gestopt of gehaald.

In figuur 5.11 is te zien hoe in het ontwerp het databeheer plaatsvindt. Getoond is de situatie voor één persistente klasse, *CFactType*. Rondom deze klasse zien we een *CFactTypeTable*, een *CFactTypeDS*, een *CFactDA* en een *ConnectionManager*. De klasse *CFactTable* is een *static* klasse, dat wil zeggen dat er maar één instantie van bestaat. Alle objecten van het type *CFactType* praten dus tegen dezelfde *CFactTable*.

De *CFactTable* representeert de daadwerkelijke tabel in de database. Deze klasse is verantwoordelijk voor het zoeken en aanmaken van een *CFactType*-object, met alle waarden die daarvan bekend zijn, uit de database. Dit kan hij doen op basis van een primaire sleutel, of (indien mogelijk) op basis van andere identificerende informatie. Het *CFactType*-object dat zo wordt aangemaakt kent zijn *CFactTable*, en kan daaraan verzoeken om opgeslagen te worden. Als er een nieuw object wordt gecreëerd en opgeslagen, voegt de *CFactTable* deze toe aan de database, en synchroniseert vervolgens eventuele auto-counters.

Om dit te kunnen doen heeft de *CFactTable* de beschikking over een *CFactDS*. Dit is een subklasse van de generieke *DataSet*, die is opgebouwd volgens de structuur van de bijbehorende tabel uit de database. De *CFactDS* is de representatie van de database-tabel: door er data in te stoppen en uit te halen, en de *CFactDS* door de *CFactDA* te laten synchroniseren met de database, beheert de *CFactTable* de data in de database.

Het hier getoonde is slechts een voorbeeld, alle persistente klassen hebben hun eigen **Table*, hun eigen **DataSet* en hun eigen **DataAdapter*. Deze *DataAdapters* communiceren allemaal met de database via dezelfde *ConnectionManager*, die zorgt voor een werkende verbinding (deze weet waar de database zich bevindt en welke drivers nodig zijn om ermee te communiceren). Het databeheer is dus in strikte zin verdeeld over allerlei verschillende klassen die, op precies dezelfde manier, eenzelfde verantwoordelijkheid dragen voor hun persistente objecten: de **Table*, de **DataSet* en de **DataAdapter*.

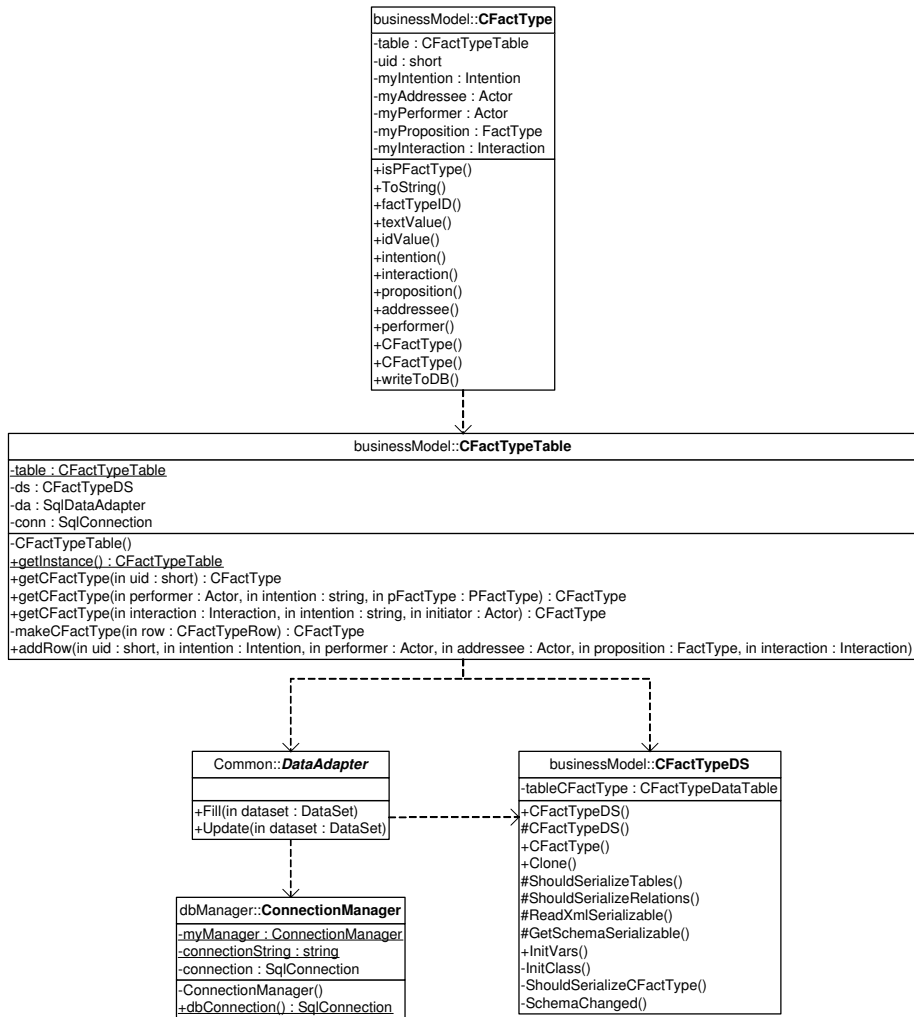
5.3 Informatiestructuur

In de voorgaande hoofdstukken hebben we eerst een bedrijfssysteem gemodelleerd, bestaande uit -onder andere- een werknemer die zijn taken bepaalt en uitvoert. Vervolgens hebben we de functionaliteit van een informatiesysteem in DEMO gemodelleerd, dat aan dat bedrijfssysteem ondersteuning kan verlenen. De functionaliteit van dat informatiesysteem wordt aangeboden door de ontworpen softwarecomponenten, die ieder een bepaalde verantwoordelijkheid uit het functionele ontwerp op zich nemen. Deze verantwoordelijkheid behelst het leveren (en het bewaren) van allerlei informatie uit het bedrijfssysteem.

Zoals we in het software-ontwerp hebben gezien, wordt voor de opslag van deze informatie gebruik gemaakt van een database. De structuur van die database, de structuur van de persistente data uit het informatiesysteem, vormt het tweede onderdeel van het technisch ontwerp. Deze informatiestructuur beschrijven we eerst op een abstract niveau, afgeleid van de informatiestructuur van het bedrijfssysteem, in een ORM-model. Vervolgens zetten we dat model om in een ERD-model, dat de structuur van de database beschrijft.

5.3.1 ORM-model

Het model is voor de overzichtelijkheid gesplitst in figuur 5.12 en figuur 5.13. In appendix A is, in figuur A.2, het hele ORM-model weergegeven. Het getoonde ORM-model is gebaseerd op het feitenmodel uit het bedrijfssysteem (zie figuur A.1 in appendix A). Dit komt doordat, zoals gezegd, de taak van het informatiesysteem bestaat uit het verschaffen (en bewaren) van informatie aan het bedrijfssysteem. De informatie die opgeslagen en bijgehouden moet worden is dus (tot op zekere hoogte) de informatie die in het bedrijfssysteem gebruikt wordt.



Figuur 5.11: Klassen voor het databaseer

In figuur 5.12 zien we het gedeelte van het ORM-model dat zich op het typeniveau bevindt, aangevuld met de concepten *Proposition*, *PFact* en *RoleInstance*. In vergelijking met het feitenmodel van het bedrijfssysteem komen een aantal concepten niet meer voor, en zijn er een aantal nieuwe concepten bijgekomen.

- *ActionRule*: de actieregels van actoren kunnen in de Designer (nog) niet gemodelleerd worden, hetgeen betekent dat de ActieManager de actieregels niet kan gebruiken. In de actieregels staan ten eerste de coördinatieve feiten genoemd die voor een actor als agendum gelden; dat zijn immers juist die coördinatieve feiten waarvoor hij een actieregel kent, het zogenaamde *entry-point* van de actieregel. Ten tweede staan in de actieregels de mogelijke reacties beschreven van de actor, de zogenaamde *exit-points*. En tenslotte staat in de actieregels voor elk *entry-point* precies beschreven welk *exit-points* moet worden gekozen, en dus welke reactie de actor geeft, afhankelijk van allerlei feiten.

Aangezien de actieregels niet in de Designer gemodelleerd kunnen worden, is de actor *select action rule* niet door het informatiesysteem te vervullen, zoals bij het functioneel ontwerp is besproken. Het informatiesysteem moet de gebruiker wel een overzicht geven van de mogelijke reacties op een bepaald agendum, dus de mogelijke *exit-points* van de bijbehorende actieregel. Aangezien ook die *exit-points* niet bekend zijn, gaat het informatiesysteem uit van de algemene regels in de DEMO-theorie: na een *request* kan de actor een *promise* doen of een *reject*.

De actieregels, hun verband (als *entry-* of *exit-point*) met verschillende coördinatieve feiten, de actoren waarvan ze deel uitmaken, en de inhoud ervan, komen dus in de ActieManager, en in de informatiestructuur ervan, niet voor.

- *RoleInstance*: een concept dat in het bedrijfssysteem niet voorkomt is de *RoleInstance*, en de daaraan verbonden feittypen, zoals '*RoleInstance has StringValue*' en '*RoleInstance has IntValue*'. Deze feittypen zijn nodig om de waarden van *RoleInstances* in de database te kunnen opslaan.

Een *Role* is één rol uit een relatie die in zijn geheel een *PFactType* vormt. Dit verband blijkt uit '*Role is part of PFactType*'. Een *PFact* is een instantie van een *PFactType*. Een *RoleInstance* is een rol in een *PFact*, dus de instantie van een *Role*. Een *RoleInstance* heeft altijd een bepaald waarde waarmee het geïnstantieerd wordt.

Stel dat er een *PFactType* bestaat, '*Persoon <1> heeft voornaam <2>*'. Een instantie hiervan zou bijvoorbeeld kunnen zijn het *PFact* '*Persoon #124483 heeft voornaam Joop*', maar ook het *PFact* '*Persoon #540009 heeft voornaam Nellie*'. Hierin zijn in totaal vier *RoleInstances* te herkennen: twee instanties van rol <1>, met de waarden #124483 en #540009, en twee instanties van rol <2>, met de waarden *Joop* en *Nellie*.

Om de waarden van de verschillende *RoleInstances* in de database op te kunnen slaan, zouden we gebruik kunnen maken van een generiek feittype '*RoleInstance has Value*'. Het datatype van het veld *Value* is echter afhankelijk van de rol die door de specifieke *RoleInstance* gespeeld wordt: in bovenstaande voorbeeld zijn twee *RoleInstances* gevuld met tekstuele waarden, hetgeen in een database bijvoorbeeld gerepresenteerd zou worden door het datatype string of varchar.

De twee andere *RoleInstances* zijn echter gevuld met numerieke waarden, hetgeen in een database bijvoorbeeld gerepresenteerd zou worden door het datatype (un)signed integer.

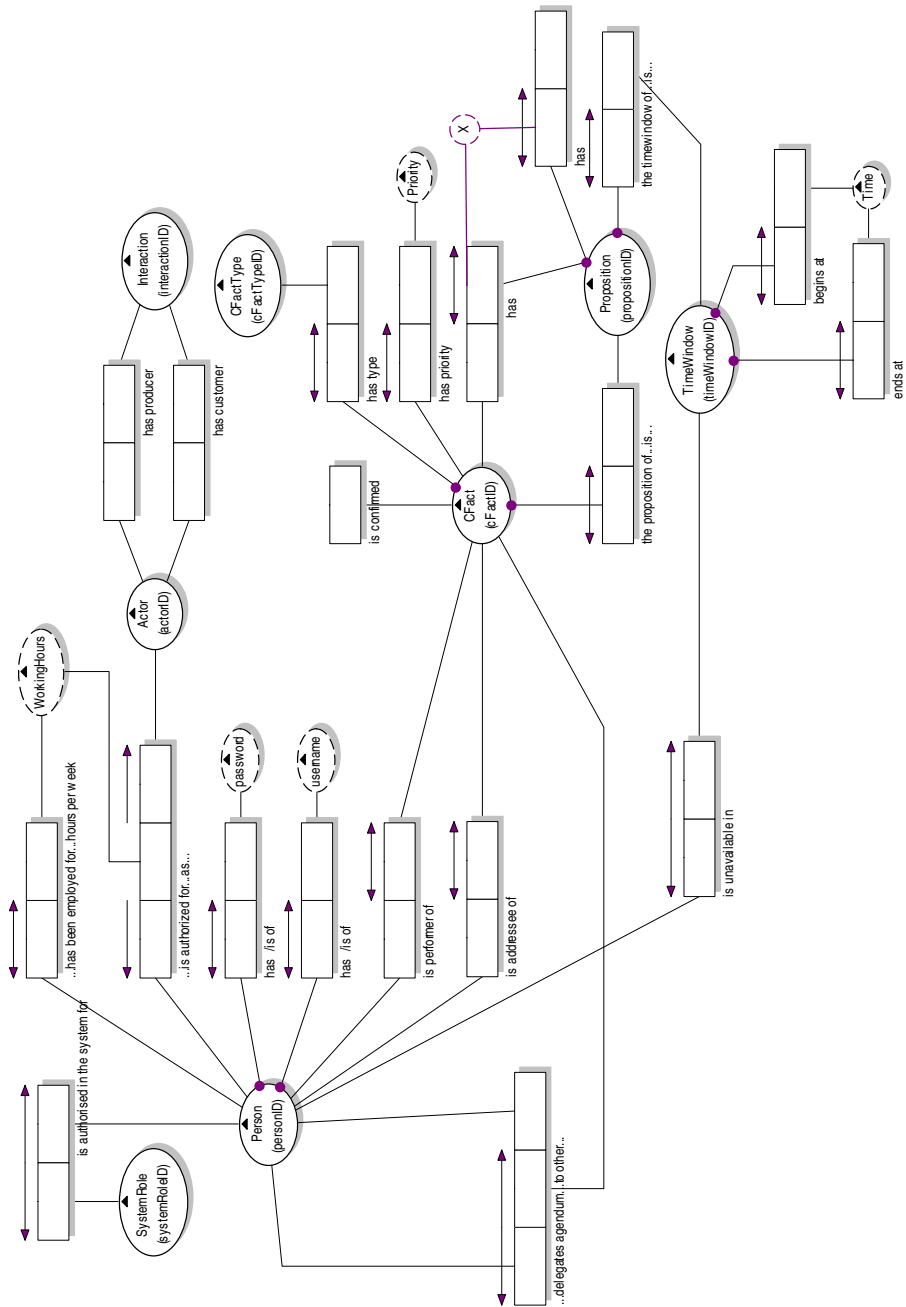
Het databaseveld waarin de waarde van een *RoleInstance* wordt opgeslagen is dus van een steeds verschillend datatype. Er bestaan twee mogelijkheden om deze informatie, met verschillende datatypen, te kunnen bewaren. Ten eerste zou gebruik gemaakt kunnen worden van een veld met het datatype *Variant*. In dit veld kan allerlei informatie van allerlei verschillende datatypen opgeslagen worden. Wij hebben ervoor gekozen om zelf vier verschillende soorten velden te maken, waarin waarden van een *RoleInstance* opgeslagen kunnen worden, namelijk voor tekstuele, numerieke, logische en temporele waarden. De keuze voor deze oplossing is voornamelijk ingegeven doordat dit in de software makkelijker is te gebruiken.

- *SystemRole*: zoals in het functionele ontwerp is beschreven hebben sommige gebruikers van het systeem extra bevoegdheden. Een normale gebruiker hoeft alleen zijn takenlijst te zien en daarop reageren. Een aantal taken, zoals het autoriseren van personen voor actorrollen, of het onderhouden van de database, zijn voorbehouden aan geprivilegeerde gebruikers. Om dit mogelijk te maken is de functionaliteit van het systeem verdeeld over een aantal systeemrollen, zoals *Worker*, *SystemManager*, *PersonnelManager* en *TimeManager*. De relatie ‘*Person is authorised for SystemRole*’ bepaalt welke mogelijkheden gebruikers hebben in het systeem. Logischerwijs zouden gebruikers voor minimaal één systeemrol geautoriseerd moeten worden, aangezien ze anders niets kunnen in het systeem.
- *Password*: zoals in de vorige sectie bij de bespreking van de gebruikersinterfaces is gezegd, is het noodzakelijk om gebruikers te kunnen identificeren. Hiervoor krijgen alle gebruikers een gebruikersnaam en wachtwoord, die in de database opgeslagen moeten worden. Indien gewenst kan in plaats van een eigen authenticatiesysteem ook gebruik worden gemaakt van bestaande authenticatiesystemen.

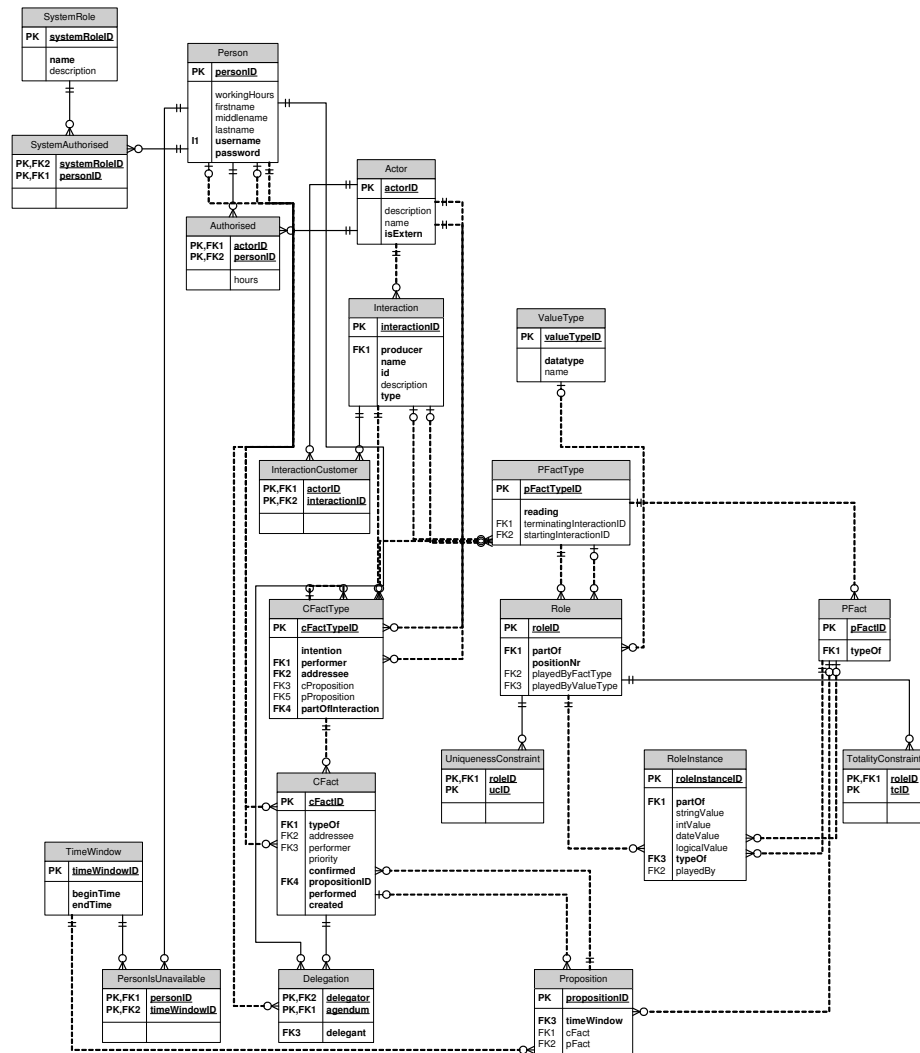
5.3.2 ERD-model

Het ORM-model is een conceptueel schema van de informatie die moet worden opgeslagen. Dit schema is om te zetten in een relationeel model, waarmee een daadwerkelijke database kan worden gebouwd. Deze omzetting verloopt met behulp van het *relational mapping*-algoritme [19], dat objecttypen met bijbehorende feittypen groepeerd tot tabellen. Dit algoritme een ORM-model om te zetten in een relationeel schema is gebaseerd op een aantal eenvoudige regels, en een aantal uitzonderingen daarop. Voor de precieze beschrijving van de omzetting verwijzen we naar [19], in het kort komt de omzetting hierop neer:

Eerst worden voor alle objecttypen een tabel gevormd, met uitzondering van de objecttypen die een rol spelen in slechts één feittype. Feittypen met een enkele uniciteitsregel (die dus een één-op-meer relatie vormen) worden in de tabel ‘getrokken’ die dat objecttype representeert, met een verwijzing naar de tabel van het andere objecttype. Voor feittypen die een uniciteitsregel hebben over meerdere rollen wordt een koppeltabel gemaakt. De primaire sleutel van de tabellen bestaat uit die kolommen die de rollen representeren die participeren in een uniciteitsregel. Het resultaat van deze omzetting is te zien in figuur 5.14; met dit relationele schema is de database geconstrueerd.



Figuur 5.13: Informatiestructuur instantieniveau



Figuur 5.14: Databaseschema ActieManager

5.4 Implementatie: beschrijving en screenshots

Het technisch ontwerp is geïmplementeerd in een prototype. Dit prototype dient om aan te tonen dat implementatie van het ontwerp mogelijk is, en als basis voor verdere productontwikkeling. De functionaliteit van het prototype zal kort beschreven worden, waarna we beschrijven wat er in een verdere ontwikkeling zou moeten gebeuren.

Hoofdmenu: de gebruiker begint de ActieManager met een hoofdscherm, waarin wat uitleg staat over het prototype en een menu dat toegang biedt tot allerlei functionaliteit. Dit scherm is getoond in figuur 5.15. De keuzemogelijkheden in het getoonde menu, en dus de functionaliteit die de ActieManager aan de gebruiker biedt, is afhankelijk van de systeemrollen die hij vervult: het menuitem *Personeelsmanagement* bijvoorbeeld is alleen beschikbaar voor gebruikers die de systeemrollen *Employ* of *Authorize* spelen.

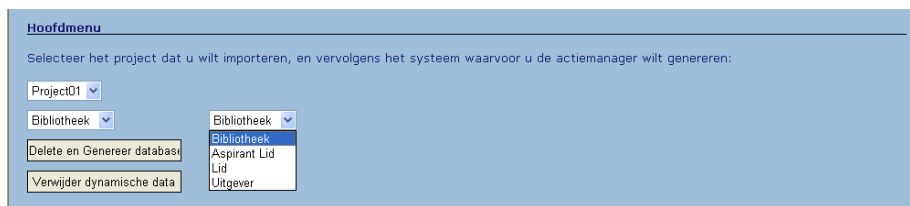


Figuur 5.15: Hoofdscherm ActieManager

Databasebeheer: indien de gebruiker de rol van systeembeheerder mag spelen, dan kan hij naar het scherm *Databasebeheer* gaan, dat in figuur 5.16 weergegeven is. In dit scherm staan alle projecten getoond die in de Designer database beschikbaar zijn. De gebruiker kan het gewenste project selecteren, waarin dus het DEMO-model is gemaakt dat hij wil importeren. Vervolgens worden alle systeemactoren getoond die in dit project aanwezig zijn, de gebruiker kan kiezen vanuit wiens 'oogpunt' hij de ActieManager wil laten werken. Een demo-model bevat namelijk geen perspectief, alle systeemactoren zijn extern ten opzichte van elkaar. Door te selecteren welke systeemactor ondersteund moet worden, geeft de gebruiker aan dat de rest van de systeem-

actoren als externe actoren beschouwd moeten worden. Vervolgens kan de gebruiker opdracht geven om het gekozen model te importeren. Hiermee krijgt de ActieManager kennis over de te ondersteunen organisatie.

Ter illustratie is in de Designer een voorbeeldmodel opgesteld van de bibliotheek-casus (het coördinatiemodel en feitenmodel daarvan zijn eerder getoond in de figuren 5.6 en 5.7), en in de ActieManager geïmporteerd. De rest van de screenshots tonen dus allen een ActieManager waarin de bibliotheekcasus is geïmporteerd.



Figuur 5.16: DatabaseBeheer ActieManager

Personeelsmanagement: na het importeren van een DEMO-model moeten de autorisaties daarvan ingesteld worden: er moet aan de ActieManager verteld worden welke personen uit de organisatie welke actorrollen mogen spelen. In het scherm *Personeelsmanagement*, dat in figuur 5.17 is getoond, is dit mogelijk. Voordat personen geautoriseerd kunnen worden, moet echter bij het systeem bekend zijn dat deze personen in dienst zijn getreden. In dit scherm is daarom allereerst mogelijk om aan te geven dat personen in en uit dienst zijn getreden, en zijn de bijbehorende persoonlijke gegevens in te vullen.

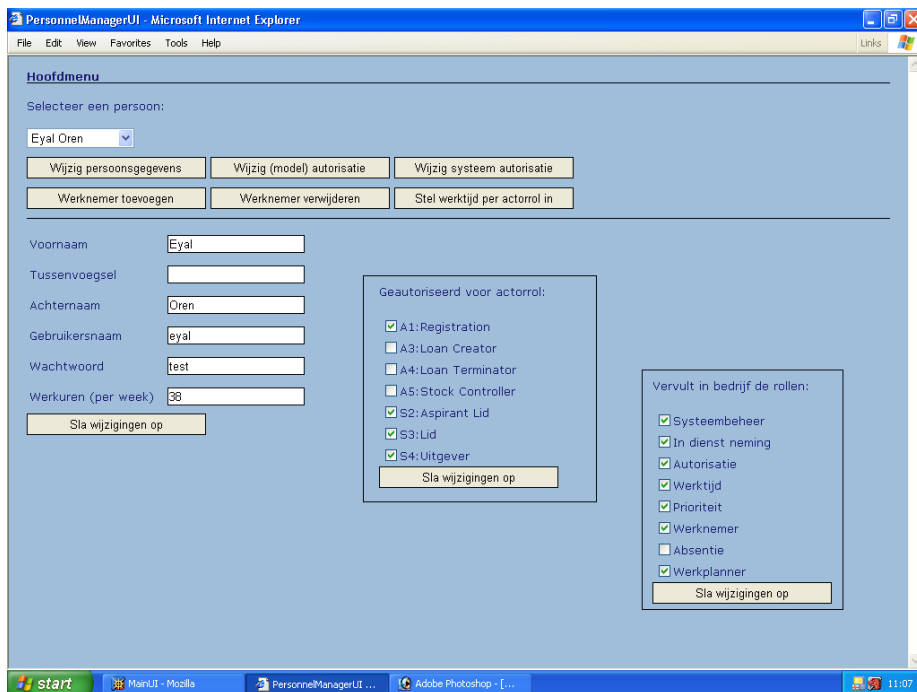
Daarnaast is het mogelijk om van elke werknemer aan te geven welke actorrollen (uit het organisatiemodel) hij mag spelen; de actorrollen die daarbij geselecteerd kunnen worden, zijn afkomstig uit het geïmporteerde organisatiemodel. Zo is in dit screenshot te zien dat de geselecteerde werknemer is geautoriseerd voor de actorrollen *Registratie*, *Aspirant lid*, *Lid* en *Uitgever*. Deze actorrollen zijn afkomstig uit het coördinatiemodel van de bibliotheekcasus.

Ook is het mogelijk om van de werknemers aan te geven voor welke systeemrollen ze geautoriseerd zijn; dit bepaalt zoals gezegd de mogelijkheden die zij ten aanzien van het systeem hebben. Gebruikers die de rol *In dienst neming* spelen mogen slechts gebruikers toevoegen en verwijderen uit het systeem. Gebruikers die de rol *Autorisatie* mogen spelen, kunnen vervolgens van personen die in dienst zijn de autorisaties instellen. Gebruikers die, zoals in dit screenshot bijvoorbeeld het geval is, beiden rollen mogen spelen, kunnen aangeven dat iemand in dienst is gekomen, en vervolgens de autorisaties van diegene instellen.

Als laatste is het in dit scherm mogelijk (hetgeen niet op het screenshot is weergegeven) om van een gebruiker aan te geven, welk deel van zijn werktijd hij aan een bepaalde actorrol is toegewezen. Als een werknemer bijvoorbeeld zowel de rol van *Registratie* als van *Loan Creator* vervult, is het mogelijk om aan te geven dat hij de rol van *Registratie* voor 10 uur per week vervult, en de rol van *Loan Creator* voor 32 uur per week. Deze informatie wordt later gebruikt bij het scheduleren van het werk.

Tijdsmanagement: voor het scheduleren van het werk is het niet alleen noodzakelijk om te weten hoeveel tijd een werknemer beschikbaar is voor zijn actorrollen, maar ook

5.4. IMPLEMENTATIE: BESCHRIJVING EN SCREENSHOTS

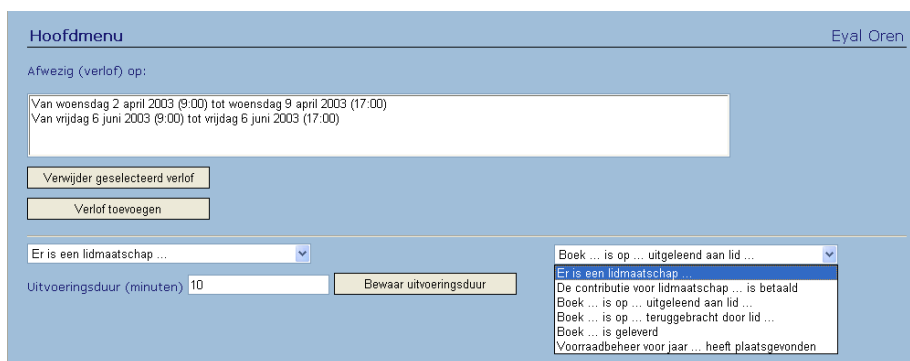


Figuur 5.17: Personeelsmanagement ActieManager

hoeveel tijd de werknemer al bezet is met het uitvoeren van bepaalde werkzaamheden. Om deze informatie aan gebruikers te kunnen bieden kan in het scherm *Tijdsmanagement* (zie figuur 5.18) worden aangegeven hoelang de productieve acties van transacties gemiddeld duren. Het instellen van deze tijdsduren is slechts mogelijk voor gebruikers die de rol van *Jobplanner* mogen spelen.

De duur van een productieve actie wordt als volgt gebruikt om een schatting te maken van de bezetting van een persoon: indien een gebruiker, in een bepaalde actorrol, de totstandkoming van een transactie heeft beloofd, maar nog niet heeft verklaard, dan betekent dit dat hij in de executiefase daarvan zit. Aangezien de executiefase bestaat uit het uitvoeren van de bijbehorende productieve actie, is de persoon -naar schatting- voor de duur van de productieve actie bezig met de executiefase van deze transactie. De totale bezetting van deze persoon (in de vervulling van deze actorrol) bestaat uit de som van de tijdsduren van alle de productieve acties, waarvan hij de totstandkoming wel heeft beloofd maar nog niet heeft verklaard. Doordat ook de capaciteit, die deze persoon voor deze bepaalde actorrol beschikbaar heeft, bekend is (dat is door de personeelmanager ingevuld), is een schatting te geven van de tijd die deze persoon nog beschikbaar heeft voor het spelen van deze actorrol.

Daarnaast is in dit scherm door gebruikers aan te geven dat ze -om welke reden ook- afwezig zijn op een bepaalde tijd. In het prototype kunnen alle gebruikers dit zelf doen, in het ontwerp staat dat dit voorbehouden moet zijn aan de zogenaamde *AbsentieManager*, een systeemrol die het verlof van werknemers moet goedkeuren en aan het systeem moet doorgeven. De afwezigheid van werknemers wordt nog niet meegenomen bij de scheduling, dit moet in de toekomst wel gebeuren.



Figuur 5.18: Tijdsmanagement ActieManager

Externe feiten: in een DEMO-model staan vaak verwijzingen naar externe feitenbanken. In deze externe feitenbanken staan feiten die niet in deze organisatie tot stand zijn gekomen, en waarvan de transacties waardoor deze feiten tot stand zijn gekomen over het algemeen ook niet bekend zijn. Deze feiten worden door actoren in de organisatie gebruikt; zij komen deze feiten te weten door middel van informatieve transacties met de beheerders van deze feitenbanken.

In de bibliotheekcasus zijn de personen die lid willen worden externe feittypen: het tot stand komen van een persoon is geen feittype dat binnen de bibliotheek plaatsvindt. Het opnemen van een verwijzing naar een externe feitenbank geeft aan dat de bibliotheek, bij het aanmaken van een lidmaatschap, persoonsgegevens opzoekt in een of andere feitenbank (bijvoorbeeld de Gemeentelijke Basis Administratie) waarin deze persoonsgegevens bewaard worden.

In het prototype van de ActieManager is het niet mogelijk om een daadwerkelijke koppeling aan te leggen met een externe feitenbank. Daarentegen kan de systeembeheerder, via het scherm *Externe feiten* (zie figuur 5.19) de externe feiten zelf invullen. In dit scherm staat een keuzelijst met alle in het feitenmodel voorkomende externe feittypen; in dit geval is dat slechts het feittype 'Er is een persoon P', de rest van de feittypen worden tot stand gebracht door transacties binnen de organisatie. Na het selecteren van het gewenste feittype is het mogelijk om feiten van dat type toe te voegen en te verwijderen. Ook toont de ActieManager de inhoud van de reeds bestaande externe feiten. Feitelijk bestaat deze oplossing er uit dat de systeembeheerder het relevante deel van de externe feitenbank met de hand in de ActieManager invoert. Bij verdere ontwikkeling zou een koppeling moeten worden gelegd met een daadwerkelijke externe feitenbank.

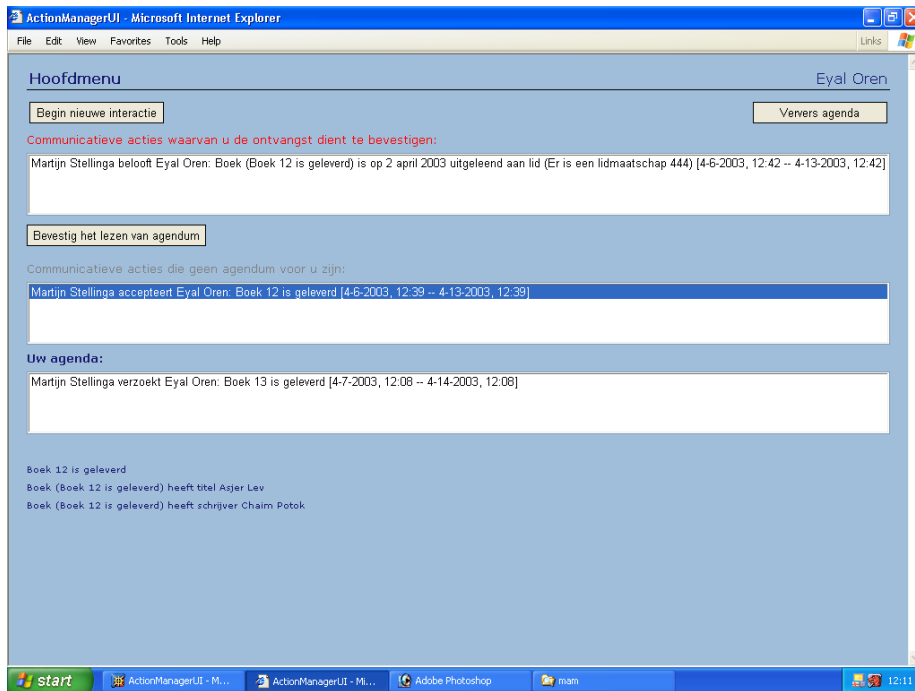
Agenda: het agendascherm is vergelijkbaar met de todo-list van andere *wfm* systemen: het toont een overzicht van de coördinatieve feiten die voor de gebruiker van belang zijn. In figuur 5.20 is een voorbeeld getoond van dit scherm. Op het scherm worden drie lijsten getoond met coördinatieve feiten: ten eerste de coördinatieve feiten waarvan deze gebruiker, als adressee, de ontvangst moet bevestigen. Een coördinatief feit bestaat volgens DEMO pas als de adressee heeft bevestigd de inhoud ervan te hebben begrepen.

Daaronder staan coördinatieve feiten die geen agendum vormen voor deze gebruiker, maar die wel voor hem van belang zijn. Een voorbeeld daarvan is een belofte die de executor van een transactie uitsprekt tegen de initiator van die transactie: de belofte is een agendum voor de executor (die moet als reactie daarop namelijk een productieve

5.4. IMPLEMENTATIE: BESCHRIJVING EN SCREENSHOTS



Figuur 5.19: Externe feitenscherf ActieManager



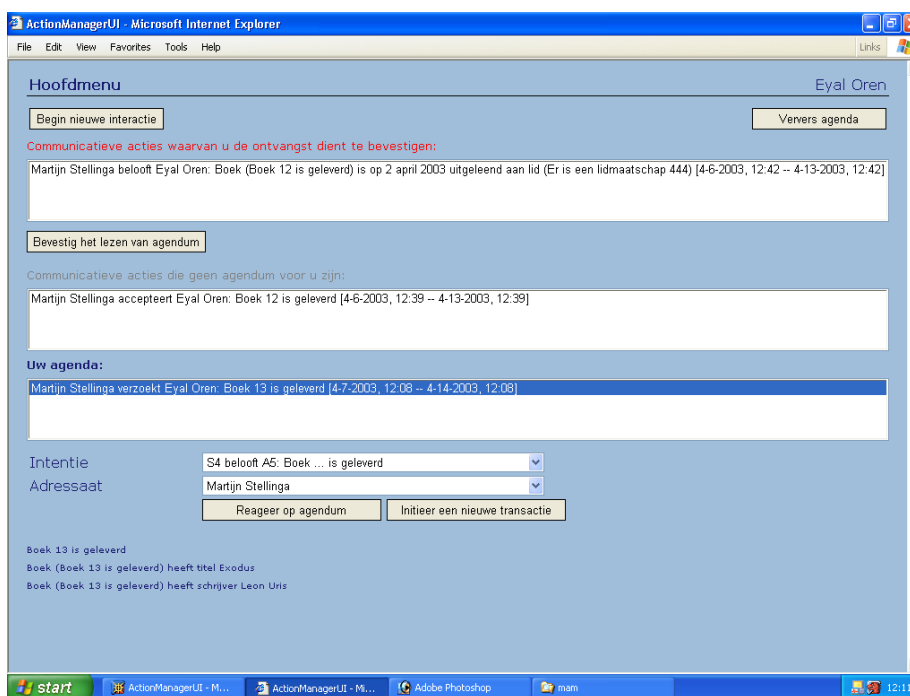
Figuur 5.20: Agendascherf ActieManager

HOOFDSTUK 5. PROTOTYPE: TECHNISCH ONTWERP EN IMPLEMENTATIE

actie uitvoeren en de tot stand koming daarvan verklaren) maar het is voor de initiator wel belangrijk om te weten dat de belofte is gedaan.

Daaronder staan de agenda van deze gebruiker. Dit zijn alle coördinatieve acties die voor hem een agendum zijn, en waarop hij dus op een of andere manier moet reageren. Indien de gebruiker één van de agenda selecteert, worden de reactiemogelijkheden getoond die hij heeft op dat agendum, hetgeen in figuur 5.21 is getoond. In dit voorbeeld zou de gebruiker op het geselecteerde agendum kunnen reageren door de propositie te beloven, of te weigeren. Ook kan de gebruiker bij het afhandelen van een agendum een nieuwe transactie initiëren, bijvoorbeeld als tijdens het aanvragen van een lidmaatschap eerst de contributie van dat lidmaatschap moet worden betaald. Dit verloopt gelijk aan een nieuwe initiatie (niet als reactie op een agendum), zie hieronder.

Als de gebruiker één van de coördinatieve feiten uit deze drie lijsten selecteert wordt de inhoud van de propositie (tot één hiërarchisch niveau diep) onderin het scherm getoond. In dit scherm is bijvoorbeeld de acceptatie van een boeklevering geselecteerd, waarbij onderin het scherm is te zien welke feiten er bekend zijn over dat specifieke boek.

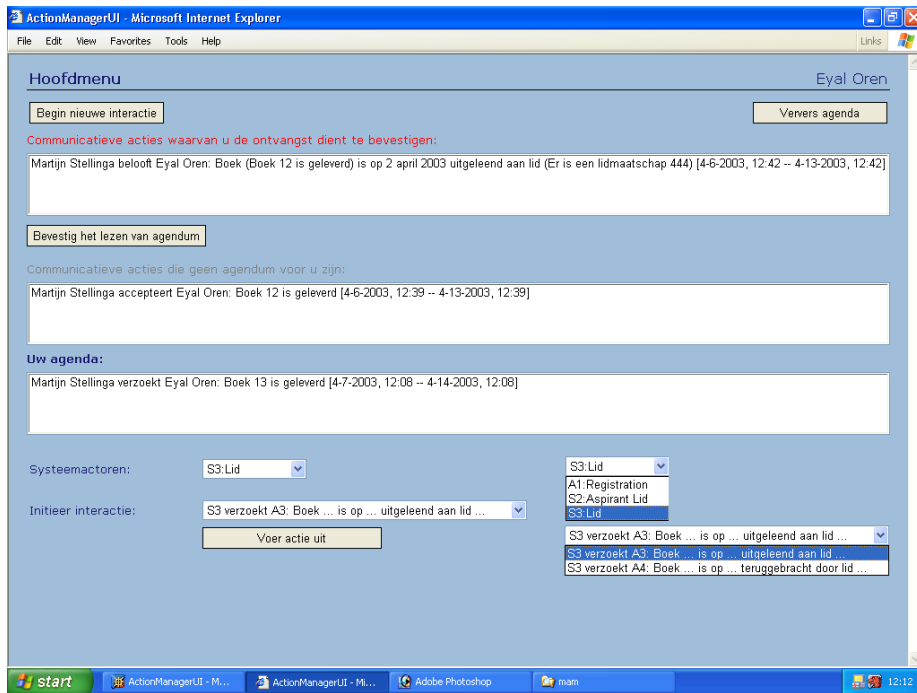


Figuur 5.21: Geselecteerd agendum ActieManager

Naast reageren op agenda is het voor gebruikers ook mogelijk nieuwe transacties te initiëren, indien het model en de autorisaties van de gebruiker dit toestaat. Als een gebruiker een externe actorrol mag vervullen dan mag hij zelfstandig interacties initiëren. Een externe actor wordt namelijk niet ondersteund in het afhandelen van zijn agenda en de reacties daarop, maar slechts in het initiëren van transacties met het ondersteunde bedrijfssysteem; externe actoren worden door de ActieManager slechts gezien als initiators van transacties met interne actoren.

Het initiëren van een transactie is in figuur 5.22 getoond. Alle actorrollen die door

5.4. IMPLEMENTATIE: BESCHRIJVING EN SCREENSHOTS



Figuur 5.22: Agendascherm ActieManager (initieëren van een interactie)

deze gebruiker gespeeld kunnen worden en die een transactie kunnen initiëren zijn getoond, en per actorrol staan de transacties genoemd die hij kan initiëren. Na het kiezen van de gewenste interactie komt de gebruiker in een ander scherm terecht waar hij de propositie kan invullen, hetgeen in figuur 5.23 is getoond. Hier kan de gebruiker allereerst de adressee selecteren, dus bepalen aan wie hij dit verzoek gaat richten. De ActieManager toont daarvoor ten eerste een lijst met alle personen die geauthoriseerd zijn om deze actorrol (de executor van deze transactie) te spelen. Ten tweede toont de ActieManager informatie waarmee de gebruiker een vorm van scheduling kan uitvoeren: onderin staat (na het selecteren van een adressee) de tijd die deze adressee voor deze bepaalde actorrol beschikbaar heeft en bezet is. Met deze informatie kan de gebruiker zelf bepalen aan wie hij zijn verzoek gaat richten.

Figuur 5.23: Propositiescherm ActieManager

Daarnaast toont de ActieManager de feitenstructuur die ingevuld moet worden bij

een initiatie. Deze feitenstructuur bestaat uit het feittype van de propositie, met eventueel daaraan middels verplichte-rol-regels verbonden andere feittypen. In dit voorbeeld wordt de transactie *Boekuitlening* geïnitieerd, waarvan het productieve feittype (zoals in het feitenmodel is gemodelleerd) is '*Boek B is op Datum D uitgeleend aan Lid L*'. Bij het invullen van dit feittype kan de gebruiker kiezen uit alle boeken en leden die er in het systeem bekend zijn. Aangezien boeken en leden tot stand komen in respectievelijk transactie T01 en transactie T05, en productieve feiten pas bestaan als de bijbehorende transactie de status *geaccepteerd* heeft bereikt, kan de gebruiker hier kiezen uit alle leden en boeken waarvan een coördinatief feit bestaat met de intentie *geaccepteerd*. Aangezien de tweede rol uit dit feittype (de datum van uitlening) gespeeld wordt door een waardetype (zie het feitenmodel), kan de gebruiker hier zelf een datum invullen; de ActieManager controleert ook of daadwerkelijk een datum is ingevoerd.

Als laatste moet de gebruiker een tijdsvenster bepalen waarbinnen hij de totstandkoming van het feit verlangt. Voor het gemak van de gebruiker wordt door de ActieManager alvast als begintijd de huidige tijd ingevuld, en de eindtijd een week later. Dit is uiteraard door de gebruiker aan te passen. Als de gebruiker de actie uitvoert komt deze bij de adressee terecht in het lijstje '*te bevestigen*', en na bevestiging in het lijstje '*agenda*'.

5.5 Aanbevelingen bij verdere ontwikkeling

Het geïmplementeerde prototype dient twee doelen: ten eerste bewijst het dat het opgestelde functionele ontwerp te implementeren valt; dat het functionele ontwerp geen onmogelijkheden bevat. Ten tweede is het een aanzet tot een commercieel systeem, indien ModulOr Technology besluit een dergelijk systeem te gaan bouwen en verkopen. Dit prototype kan op twee manieren de basis vormen van de ontwikkeling van een dergelijk commercieel systeem: ten eerste kan het ontwerp en de code van het prototype gebruikt worden, en uitgebreid tot een commercieel systeem. Ten tweede kunnen, indien er voor wordt gekozen een nieuw ontwerp te maken, enkele lessen uit het prototype getrokken worden die een nieuwe ontwikkeling moeten vergemakkelijken. Een aantal punten die belangrijk zijn bij een eventuele verdere ontwikkeling worden hieronder beschreven.

- *Gebruikerstesten*: in hoofdstuk 4 is de ontworpen functionaliteit van de ActieManager vergeleken met de functionaliteit van andere (commerciële) *wfm* systemen. Daaruit volgde de conclusie dat de ActieManager in basisfunctionaliteit niet afwijkt van andere *wfm* systemen.

In die vergelijking is niet aan bod gekomen dat er voor gebruikers grote verschillen bestaan tussen *activiteit-gerichte* en *communicatie-gerichte wfm* systemen. Het overgrote deel van de commercieel verkrijgbare *wfm* systemen is, in tegenstelling tot de ActieManager, *activiteit-gericht*. De *communicatie-gerichte* benadering van DEMO en van de ActieManager betekent voor gebruikers een andere manier van denken en werken. Juist bij *wfm* systemen, die gebruikers dienen te helpen bij de uitvoering van hun werk, is de gebruiksvriendelijkheid van het informatiesysteem, en vooral de congruentie met het mentale model van de gebruiker, van groot belang.

Daarom zijn gebruikerstesten noodzakelijk om de vraag te kunnen beantwoorden in hoeverre een op DEMO gebaseerd *wfm* systeem bruikbaar is; wellicht blijkt uit deze testen bijvoorbeeld dat gebruikers eerst een introductie in DEMO moeten

krijgen, of dat in sommige werksituaties een activiteit-gerichte benadering beter aansluit bij de werkvloer. DEMO gaat ervan uit dat communicatie-gerichte modellering per definitie beter aansluit bij de gebruikers, omdat deze modellering zich juist baseert op hoe mensen met elkaar omgaan en met elkaar samenwerken; of dit ook werkelijk zo is zal moeten blijken. Dit onderzoek heeft gekeken hoe een *wfm* systeem op basis van DEMO eruit moet zien; aanvullend onderzoek is nodig om te bekijken wat gebruikers, de werknemers, in de praktijk vinden van het werken volgens DEMO.

- *Functionaliteit*: in de functionaliteit van het prototype ontbreken een aantal elementen, die voor een commercieel systeem noodzakelijk zijn:
 - *Management rapportage*: een belangrijke functionaliteit van *wfm* systemen bestaat uit het aanbieden van allerlei informatie aan het management. Aangezien werkbesturing volledig via het informatiesysteem verloopt, bevat dat systeem allerlei informatie over de werkstromen binnen de organisatie. Analyse van die informatie kan bijvoorbeeld indicaties geven over eventuele bottlenecks in het bedrijfsproces, over- of onderbezetting van actorrollen, of prestaties van individuen en afdelingen. Een aantal van deze analyses kunnen ook ten aanzien van het organisatiemodel gedaan worden, maar werkelijke data levert uiteraard een vollediger beeld. Het aanbieden van ruwe operationele data of analyses daarvan kan daarom zeer waardevol zijn, vele commerciële pakketten bieden dit dan ook aan. In de ontwikkeling van een commercieel systeem zou in ieder geval een vorm van rapportage moeten worden opgenomen. Eventueel zou onderzoek kunnen worden gedaan naar mogelijke geautomatiseerde analyses van de vergaarde data, waarmee problemen in de organisatie kunnen worden opgespoord.
 - *Niet-afgehandelde agenda*: in DEMO wordt ervan uitgegaan dat actoren hun verantwoordelijkheden nemen en daarnaar handelen. Mensen gedragen zich niet altijd als ideale actoren, en komen niet altijd hun afspraken na; ze handelen dus niet altijd al hun agenda af. Om hierop te kunnen inspelen, is het wenselijk dat het *wfm* systeem allerlei overzichten kan geven van niet-afgehandelde agenda. Een voorbeeld is een overzicht van alle beloofde proposities waarvan het tijdsvenster is verlopen; dit betekent immers dat iemand iets heeft beloofd tot stand te brengen, maar dat vervolgens niet heeft gedaan. Een ander voorbeeld is een verzoek waarop niet gereageerd wordt: een actor is verantwoordelijk voor het afhandelen van een verzoek (met een belofte of een afwijzing); niet reageren op een agendum is het negeren van de verantwoordelijkheid.
Een overzicht van deze niet-afgehandelde agenda is een bijzonder geval van management rapportage: voor het management is het belangrijk om dergelijke informatie te kennen, aangezien het aangeeft dat bepaalde werknemers hun verantwoordelijkheden niet goed dragen, om welke reden dan ook.
 - *Koppelingen met externe applicaties*: koppelingen met externe applicaties zijn in het geheel buiten dit onderzoek gelaten. Deze koppelingen zouden bijvoorbeeld kunnen bestaan uit het opstarten van externe applicaties vanuit het *wfm* systeem of het gebruik van documentaire informatiesystemen om allerlei documenten die samenhangen met transacties aan gebruikers te tonen. In een commercieel systeem zou aandacht gegeven moeten worden

aan de vraag welke type koppelingen onontbeerlijk zijn en hoe dit technisch gerealiseerd zou moeten worden.

- *Aanpassen propositie*: in het prototype kan een executor een verzoek van een initiator afwijzen, zoals de DEMO-theorie voorschrijft. De initiator kan dat accepteren, en stoppen met de transactie, of hij kan nogmaals verzoeken aan de executor om de propositie tot stand te brengen. In het huidige prototype kan hij bij het herhalen van zijn verzoek niets aan de propositie aanpassen. Dit is niet erg nuttig: als de executor het verzoek om een propositie tot stand te brengen heeft afgewezen dan is de kans niet groot dat hij, bij een tweede verzoek, het tot stand brengen van precies dezelfde propositie wel zal beloven.

Het opnieuw verzoeken van een propositie heeft wel nut in het geval dat er (kleine) details van de propositie zijn gewijzigd. Als de initiator bijvoorbeeld het tijdsvenster van de propositie heeft verlengd of een iets ander model fiets verzoekt.

De mogelijkheid om details van de propositie aan te passen in het geval van een hernieuwd verzoek is niet geïmplementeerd. Dit is echter niet moeilijk te doen, en zou zeker moeten gebeuren indien tot ontwikkeling van een commercieel product wordt overgegaan.

- *Software*: de software van het prototype is uiteraard, door het verschil in doel, toepassing en ontwikkeltijd, van een ander niveau dan software van een commercieel systeem zou moeten zijn. Een aantal punten waarmee bij de verdere ontwikkeling rekening gehouden zou moeten worden:
 - *Database transaction-management*: huidige database management systemen bieden over het algemeen de mogelijkheid om database opdrachten in transacties te vatten. Deze transacties bestaan uit een aantal opdrachten die atomair kunnen worden uitgevoerd. Dit betekent dat als één opdracht uit de transactie faalt, de gehele transactie teruggedraaid kan worden. Hiermee worden inconsistente toestanden van de database vermeden. Van deze mogelijkheid is vanwege de beperkte tijd in het prototype geen gebruik gemaakt; deze mogelijkheid voegt immers niets toe aan de functionaliteit van het prototype, maar helpt om een correcte werking van het systeem te bereiken. In een commercieel systeem zou het verstandig zijn deze mogelijkheid toe te passen.
 - *Persistente objecten*: aangezien de internet-server elk verzoek als afzonderlijke aanvraag beschouwd, moeten alle objecten telkens opnieuw opgebouwd worden. Dit betekent dat bij elke aanvraag de benodigde data in de database wordt opgevraagd, en de benodigde objecten aangemaakt worden. Dit is niet efficiënt, en wellicht kan het op een andere manier opgelost worden, zonder het object-georiënteerde paradigma te verlaten. Door het object-georiënteerde paradigma (deels) te verlaten, en geen objecten te gebruiken, is het probleem uiteraard ook opgelost, maar met alle nadelige gevolgen van dien.

Dit probleem is geen onderdeel van dit onderzoek, en zou tijdens de ontwikkeling van een commercieel systeem eventueel aangepakt kunnen worden door iemand met ervaring in het ontwerpen en programmeren van internet-applicaties, en in het bijzonder met het .NET framework.

- *Foutafhandeling*: de code van het prototype is, zoals gezegd, niet geschreven met een commercieel product voor ogen. Hierdoor is op sommige plekken aan foutafhandeling niet genoeg gedaan. Dit betekent dat de gebruiker soms foutmeldingen te zien krijgt die hij niet begrijpt en dat het systeem foutmeldingen aan de gebruiker doorgeeft die het zelf zou kunnen afhandelen. Uiteraard dient dit bij de ontwikkeling van een commercieel systeem correct afgehandeld te worden.
- *Scheduling*: ten aanzien van het optimaal inplannen van resources aan taken zijn veel onderzoeksresultaten beschikbaar. In het prototype heeft het algoritme dat personen selecteert om een actorrol te vervullen niet voldoende aandacht gekregen, en onderzoeksresultaten uit bijvoorbeeld Operations Research zijn niet gebruikt. Beschikbare onderzoeksresultaten kunnen worden gebruikt om de scheduling binnen de ActieManager te verbeteren. Wellicht is het niet haalbaar om de scheduling van de ActieManager om een niveau te brengen als bij activiteit-gerichte benaderingen, aangezien daar de focus juist sterk op taken en resources ligt, maar verbeteringen zijn zeker te behalen.
- *Gebruiksvriendelijkheid*: de gebruikersinterfaces van het prototype zijn niet ontworpen voor de toekomstige gebruikers maar om de functionaliteit te testen. In het algemeen zullen daarom delen van de gebruikersinterfaces herontworpen moeten worden. Drie belangrijke punten van verbetering zijn in ieder geval:
 - *'Shortcuts' van handelingen*: het is voor gebruikers handig als bepaalde handelingen die vaak gezamenlijk worden uitgevoerd, met één zogenaamde 'shortcut' zijn uit te voeren. Een voorbeeld hiervan is het bevestigen van een coördinatief feit, en het uitvoeren van een reactie daarop. Hiervoor zijn in de huidige gebruikersinterface twee handelingen nodig (eerst het bevestigen, en daarna het reageren). Aangezien deze twee handelingen vaak direct na elkaar plaatsvinden, zou het handig zijn om ook een mogelijkheid te maken om in één handeling een coördinatief feit te bevestigen en daar direct op te reageren.

Een ander voorbeeld is het afhandelen van verzoeken door de executor: het zou voor de gebruikers handig zijn als een executor op een verzoek, in plaats van eerst te beloven en vervolgens te verklaren, direct kon verklaren dat de propositie tot stand is gebracht. Dit scheelt één handeling, hetgeen voor transacties met een eenvoudige, snelle, productieve actie vaak handig is.
 - *Overzicht van taken*: een actor heeft in de DEMO-theorie slechts behoefte aan het kennen van zijn agenda. In de werkelijkheid is het voor veel mensen ook prettig om het resultaat van bepaalde coördinatieve acties te weten, ondanks het feit dat deze geen agendum voor hen vormen, en hiervan ook een overzicht te hebben. De belofte van een executor om een propositie tot stand te brengen, is bijvoorbeeld geen agendum voor de initiator. Toch is het voor veel mensen prettig om een overzicht te hebben van beloftes die andere mensen aan hen hebben gedaan, al was het maar om niet te vergeten in welke status die transactie zich bevindt, en op wie ze nu wachten.

Daarom is in het prototype niet alleen een overzicht van de agenda en de te bevestigen coördinatieve feiten van een gebruiker opgenomen, maar ook

HOOFDSTUK 5. PROTOTYPE: TECHNISCH ONTWERP EN IMPLEMENTATIE

een overzicht van coördinatieve feiten die wel belangrijk zijn om te weten, maar geen agendum vormen. Dit is een non-functioneel aspect dat belangrijk is bij het gebruik. Dit aspect moet verder uitgewerkt worden. Het zou bijvoorbeeld meer gebruiksvriendelijk kunnen zijn om de drie huidige overzichten samen te voegen tot één lijst. In die ene lijst zouden kleurcodes kunnen aangeven welke elementen agenda zijn, welke elementen coördinatieve feiten zijn die bevestigd moeten worden, en welke elementen slechts ter informatie dienen.

- *Communicatie over propositie*: in de DEMO-theorie stelt de initiator een propositie op, waarvan hij de totstandkoming aan de executor verzoekt. Een verzoek dat de executor niet tot stand kan brengen zal worden afgewezen. Soms zijn een aantal details van een propositie voor de initiator niet zo van belang, terwijl hij tegelijkertijd niet zo goed weet wat voor de executor acceptabel is. De initiator kan natuurlijk een aantal keer dezelfde propositie verzoeken, en deze telkens aan te passen, totdat het verzoek door de executor niet langer wordt afgewezen. Het is echter vaak handiger als de initiator en de executor van tevoren met elkaar kunnen overleggen over een redelijke propositie. Het zou daarom zeer handig zijn als de ActieManager de mogelijkheid daartoe zou aanbieden.

Hoofdstuk 6

Conclusie

De onderzoeksvraag van dit werk is als volgt geformuleerd: hoe is een workflow management systeem ingericht dat werkt op basis van een DEMO-beschrijving van een organisatie? Om deze onderzoeksvraag te beantwoorden zijn drie deelvragen bestudeerd, achtereenvolgens: wat is een *wfm* systeem, hoe is uit een DEMO-model een functioneel ontwerp van een *wfm* systeem op te stellen, en valt een dergelijk functioneel ontwerp te realiseren?

Functionele criteria *wfm* systeem Uit het referentiemodel van de Workflow Management Coalition zijn functionele criteria opgesteld voor een *wfm* systeem. Een *wfm* systeem is volgens de Coalition “een systeem dat de uitvoering van workflows definieert, creëert en beheert, door het gebruik van software die de procesdefinitie kan interpreteren, en met gebruikers en externe applicaties kan samenwerken”, waarbij de term *workflow* is gedefinieerd als “de automatisering van (een gedeelte van een) bedrijfsproces waarbij documenten, informatie of taken volgens bepaalde procedures van de ene naar de andere deelnemer worden doorgegeven”.

Een *wfm* systeem moet ondersteuning bieden bij het modelleren van de workflow, het uitvoeren van de workflow en het samenwerken met gebruikers en externe applicaties. De workflow engine is het component dat verantwoordelijk is voor het uitvoeren van een workflow. Hiervoor interpreteert het de definitie van het bedrijfsproces, beheert het de verschillende instanties daarvan, en wijst het voor elke instantie daarvan taken toe aan resources. Daarnaast biedt het mogelijkheden voor toezicht en biedt het interfaces met externe applicaties en workflow engines.

Daarmee is de eerste deelvraag beantwoord, namelijk wat karakteristiek is aan een *wfm* systeem, of welke functionele criteria daaraan worden gesteld.

Workflow management volgens DEMO De CAP-theorie (die de theoretische basis vormt van DEMO) beschrijft wat organisaties zijn en hoe ze werken. Volgens de CAP-theorie zijn organisaties op te vatten als discrete dynamische systemen. Actoren zijn de enige actieve componenten van dit systeem, en slechts door hun acties vinden toestandsovergangen plaats. De activiteiten die actoren uitvoeren zijn beschreven in de actorcyclus, die door alle actoren telkens doorlopen wordt: een actor selecteert een af te handelen agendum uit zijn agenda, selecteert de bijbehorende actieregel, en voert de in de actieregel beschreven acties uit. Deze cyclus wordt oneindig vaak herhaald. De enige aanleiding voor een actor om acties uit te voeren zijn de agenda die voor

hem gelden en voor wiens afhandeling hij verantwoordelijk is; het beschrijven van de wijze waarop agenda tot stand komen en afgehandeld worden door actoren beschrijft daarmee volledig de werking van organisaties in termen van toestanden en toestands-overgangen.

Uit de beschrijving van de actorcyclus en aanvullende concepten uit de CAP-theorie is een generiek model opgesteld van een actor die, in samenwerking met andere actoren, zijn verantwoordelijkheden binnen een organisatie nakomt. Dit model beschrijft de innerlijke werking van elke willekeurige actor. Volgens de CAP-theorie beschrijft het hiermee op een abstracte wijze de werking van elke willekeurige organisatie, aangezien de actoren de enige actieve componenten daarvan zijn.

Personen die werkzaam zijn in een organisatie vervullen één of meer actorrollen, en handelen steeds in één van de aan hen toegewezen actorrollen. Het model beschrijft dus ook hoe medewerkers van een organisatie, handelend in hun actorrollen, samenwerken en hun verantwoordelijkheden dragen. Dit model is te beschouwen als een bedrijfssysteem, een sociaal systeem op het hoogste abstractieniveau van DEMO.

Functioneel ontwerp van een ondersteunend informatiesysteem Sommige actoren uit dit gemodelleerde bedrijfssysteem hebben informatie nodig voor de uitvoering van hun werkzaamheden. Het bedrijfssysteem wordt daarom per definitie ondersteund door een informatiesysteem dat informatie levert aan (en bewaart voor) het bedrijfssysteem. De functionele eisen voor een informatiesysteem dat deze ondersteuning biedt zijn vastgesteld door eerst een deelverzameling op te stellen van de informatiebehoefte uit het bedrijfssysteem¹. Uit deze informatiebehoefte is vervolgens afgeleid welke informatie door het informatiesysteem moet worden bewaard, namelijk die informatie waaraan behoefte is maar die buiten het informatiesysteem tot stand komt. Tezamen vormen deze het functionele ontwerp van het informatiesysteem.

Dit functionele ontwerp is vergeleken met de (van de Workflow Coalition afgeleide) functionele criteria voor *wfm* systemen. Het functionele ontwerp voldoet daaraan, met uitzondering van twee punten: het aanbieden van interfaces met externe applicaties en workflow engines en het aanbieden van mogelijkheden om toezicht uit te oefenen. Verder biedt het functionele ontwerp alle mogelijkheden die een *wfm* systeem volgens de Workflow Coalition zou moeten aanbieden.

De interfaces met externe applicaties en workflow engines zouden zeker toegevoegd kunnen worden, maar zijn vanwege tijdsnood niet in het project opgenomen. Hetzelfde geldt voor de mogelijkheden om toezicht uit te oefenen. In paragraaf 5.5 is beschreven welke functionaliteit aan het ontwerp zou moeten worden toegevoegd om een commercieel product te kunnen leveren, dat de kwalificatie *wfm* systeem zou kunnen dragen.

Daarmee is de tweede deelvraag beantwoord, namelijk hoe uit een DEMO-model een functioneel ontwerp is op te stellen van een *wfm* systeem. Het antwoord is dat een dergelijk functioneel ontwerp niet specifiek is voor een organisatie, en dus hoeft te worden afgeleid van het specifieke DEMO-model van die organisatie. Het functionele ontwerp is af te leiden uit de CAP-theorie, die beschrijft hoe organisaties werken en dus welke informatiebehoefte daarin bestaat. Doordat het functionele ontwerp gebaseerd is op de CAP-theorie maakt het gebruik van concepten uit de DEMO-methodiek om organisaties te beschrijven, en werkt het dus op basis van DEMO-modellen van organisaties. Het functionele ontwerp valt te classificeren als een niet-volledig *wfm* systeem, om-

¹De informatiebehoefte van het bedrijfssysteem die niet door het ontworpen informatiesysteem vervuld wordt blijft uiteraard bestaan, en zal door een ander informatiesysteem moeten worden aangeboden.

dat niet aan alle door de Workflow Coalition genoemde functionele criteria is voldaan. Wel is aan de meest belangrijke criteria voldaan, en zijn de noodzakelijke uitbreidingen beschreven.

Implementatie functioneel ontwerp Het functionele ontwerp is geïmplementeerd in een prototype. Bij deze implementatie zijn geen onmogelijkheden gevonden in het functionele ontwerp. Ondanks het feit dat een klein gedeelte van het functionele ontwerp niet is geïmplementeerd (het prioriteitsmanagement), valt te concluderen dat het zeer wel mogelijk is het functionele ontwerp is een werkend systeem te realiseren.

Hiermee is de derde onderzoeksvraag beantwoord, namelijk dat het functionele ontwerp valt te realiseren.

HOOFDSTUK 6. CONCLUSIE

Bibliografie

- [1] W.M.P. van der Aalst. Finding Errors in the Design of a Workflow Process: A Petri-net-based Approach. In W.M.P. van der Aalst, G. De Michelis, and C. A. Ellis, editors, *Proceedings of Workflow Management: Net-based Concepts, Models, Techniques and Tools (WFM'98)*, volume 98/7 of *Computing Science Reports*, pages 60–81, Lisbon, Portugal, 1998. Eindhoven University of Technology, Eindhoven.
- [2] W.M.P. van der Aalst. Making work flow: On the application of petri nets to business process management. In J. Esparza and C. Lakos, editors, *Application and Theory of Petri Nets 2002*, volume 2360 of *Lecture Notes in Computer Science*, pages 1–22. Springer-Verlag, 2002.
- [3] W.M.P. van der Aalst, P. de Crom, R. Goverde, K.M. van Hee, W. Hofman, H. Reijers, and R.A. van der Toorn. ExSpect 6.4: An Executable Specification Tool for Hierarchical Colored Petri Nets. In M. Nielsen and D. Simpson, editors, *Application and Theory of Petri Nets 2000*, volume 1825 of *Lecture Notes in Computer Science*, pages 455–464. Springer-Verlag, Berlin, 2000.
- [4] W.M.P. van der Aalst and K.M. van Hee. *Workflow Management: Models, Methods, and Systems*. MIT press, Cambridge, MA, 2002.
- [5] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A. P. Barros. Workflow Patterns. QUT Technical report, FIT-TR-2002-02, Queensland University of Technology, Brisbane, 2002.
- [6] K.R. Abbott and S.K. Sarin. Experiences with workflow management: Issues for the next generation. In *Proceedings of ACM CSCW'94 Conference on Computer-Supported Cooperative Work*, Workflow and Information Sharing, pages 113–120, 1994.
- [7] E. Auramäki and K. Lyytinen. On the success of speech acts and negotiating commitments. In E. Verharen, N. Van der Rijst, and J. L. G. Dietz, editors, *Proceedings of the First International Workshop on Communication Modeling, the Language/ Action Perspective (LAP'96)*, Oisterwijk, 1996.
- [8] G. Bakema, J.P. Zwart, and H. van de Lek. *Volledig communicatiegeoriënteerde informatiemodellering: FCO-IM*. Kluwer, Deventer, 1996.
- [9] Deloitte & Touche Bakkenist. *ExSpect User Manual*, 1999.
- [10] J.L.G. Dietz. Generic recurrent patterns in business processes. zie <http://www.demo.nl>.

BIBLIOGRAFIE

- [11] J.L.G. Dietz. Integrale zorg vraagt om integrale architectuur. zie <http://www.demo.nl>.
- [12] J.L.G. Dietz. *Introductie Tot DEMO*. Samson BedrijfsInformatie, Alphen a/d Rijn, 1996.
- [13] J.L.G. Dietz. The atoms, molecules and matter of organizations. In J. Barjis, J.L.G. Dietz, and G. Goldkuhl, editors, *Proceedings of the Seventh International Workshop on the Language/ Action Perspective on Communication Modeling (LAP2002)*, 2002.
- [14] J.L.G. Dietz. DEMO, from informational engineering to organizational engineering (sheets), 2002. zie <http://www.demo.nl>.
- [15] J.L.G. Dietz. Towards a discipline of organizational engineering (sheets), 2002. zie <http://www.demo.nl>.
- [16] J.L.G. Dietz. The atoms, molecules and fibers of organizations. *Data & Knowledge Engineering*, 2003.
- [17] F. Dignum, H. Weigand, and E. Verharen. Preface. In F. Dignum and J. L. G. Dietz, editors, *Proceedings of the Second International Workshop on Communication Modeling, the Language/ Action Perspective (LAP'97)*, Veldhoven, 1997.
- [18] D. Georgakopoulos, M. Hornick, and A. Sheth. An Overview of Workflow Management: From Process Modelling to Workflow Automation Infrastructure. *Distributed and Parallel Databases*, 3(2):119–153, April 1995.
- [19] T.A. Halpin. *Information Modeling and Relational Databases: From Conceptual Analysis to Logical Design*. Morgan Kaufmann, San Fransisco, 2001.
- [20] K.M. van Hee and W.M.P. van der Aalst. *Workflow Management: modellen, methoden en systemen*. Academic Service, Schoonhoven, 1997.
- [21] B.J. Hommes and J.L.G. Dietz. Business process modeling for the purpose of applying internet technology. EMMSAD'01, Interlaken, Switzerland, 2001.
- [22] S. Jablonski. Workflow management between formal theory and pragmatic approaches. *Lecture Notes in Computer Science*, 1806:345–358, 2000.
- [23] R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The action workflow approach to workflow management technology. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW'92)*, Emerging technologies for cooperative work, pages 281–288, Toronto, Ontario, 1992. ACM Press.
- [24] C. Mohan. Recent trends in workflow management products, standards and research. In A. Dogac, L. Kalinichenko, T. Özsu, and A. Sheth, editors, *Workflow Management Systems and Interoperability*, volume 164 of *NATO ASI Series*. Springer-Verlag, 1997.
- [25] V.E. van Reijswoud and J.L.G. Dietz. *DEMO Modelling Handbook, Volume I*. TU Delft, 1999. zie <http://www.demo.nl>.

- [26] E. Verbeek and W.M.P. van der Aalst. Woflan 2.0: A Petri-net-based workflow diagnosis tool. *Lecture Notes in Computer Science*, 1825, 2000.
- [27] W.S.M. Verheggen and G.A.M. Widdershoven. Informed consent: Implementing shared decision-making in health care. In E. Verharen, N. Van der Rijst, and J.L.G. Dietz, editors, *Proceedings of the First International Workshop on Communication Modeling, the Language/ Action Perspective (LAP'96)*, Oisterwijk, 1996.
- [28] WFMC. Workflow Management Coalition Terminology and Glossary. Technical Report (WFMC-TC-1011), Workflow Management Coalition, Brussels, 1999.

BIBLIOGRAFIE

Nawoord

Terugkijkend op mijn afstuderen heb ik veel geleerd. Ik heb in dit afstudeerproject ten eerste veel -voor mij nieuwe- technologieën gebruikt. Het toepassen daarvan is op zich prima gelukt: het programmeren in .NET, het werken aan een internet-applicatie, het opzetten van een internet-server, het gebruiken van een SQLServer database en stored procedures, het is allemaal goed gegaan. Het toepassen van deze technologieën heeft mij ten eerste wel meer tijd en moeite gekost dan ik van tevoren had gedacht, vooral omdat de onbekendheid veel twijfel over de gemaakte keuzes opriep. Ten tweede brengt onervarenheid altijd, ook nu, met zich mee dat problemen niet handig opgelost worden, oplossingen niet optimaal zijn, en bestaande structuren niet gezien worden. Het is bij tijden frustrerend om na vier maanden programmeren te zien hoe sommige dingen veel makkelijker opgelost hadden kunnen worden, mits bepaalde kennis op dat moment al aanwezig was geweest. Jammer genoeg was er ook binnen ModulOr Technology niet iemand die met deze specifieke technologieën ervaren was.

Ten tweede heb ik in dit afstudeerproject duidelijk gemerkt dat ik graag problemen op een theoretisch abstractieniveau beschouw, waar men zich bij ModulOr Technology naar mijn idee op een meer praktisch en commercieel niveau begeeft. Ik heb van ModulOr Technology gelukkig de vrijheid gekregen om mijn afstudeeropdracht in te vullen in een richting die ik zelf interessant vond. Het nadeel daarvan is dat daardoor de samenwerking en begeleiding niet altijd optimaal waren. Juist daarom ben ik blij dat ModulOr Technology enthousiast is over het geleverde werk, en dat het is gelukt om met een theoretische benadering vanuit de DEMO-theorie te komen tot een praktisch product waar ModulOr Technology mee verder kan.

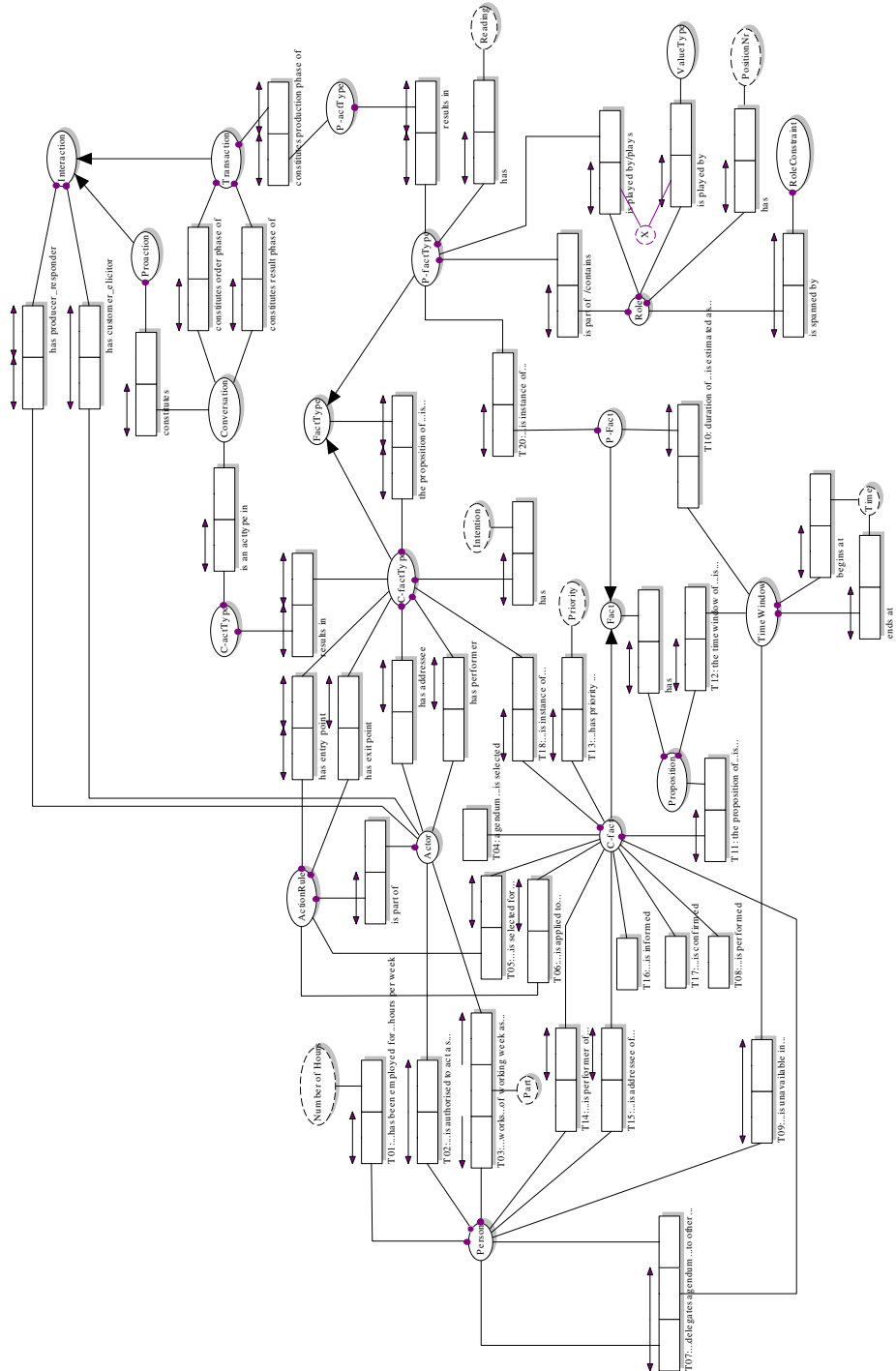
Ten derde heb ik gemerkt hoe leuk het is om van dichtbij een voortdurende wetenschappelijke ontwikkeling mee te maken: DEMO was in de beginperiode van mijn afstuderen heftig in beweging. Dit leidde ertoe dat ik met mijn mede-afstudeerder Martijn Stellinga dagelijks discussies voerde over de betekenis van bepaalde concepten en hoe dat coherent bleef met de rest van DEMO. Vooral de momenten dat Jan Dietz plotseling aangaf dat het toch anders zat, ons daarmee in verwarring, dankbaarheid of frustratie achterlatend, waren -hoe gek dat ook klinkt- juist zeer motiverend; discussies over DEMO vond ik vaak leuker dan het einddoel van het afstuderen.

Al met al kan ik zeggen dat ik veel heb opgestoken deze tijd. Ik zal veel dingen een volgende keer anders doen; de fouten die ik achteraf bij mezelf heb opgemerkt, en de problemen die ik niet had zien aankomen, zullen hopelijk lessen vormen voor de toekomst.

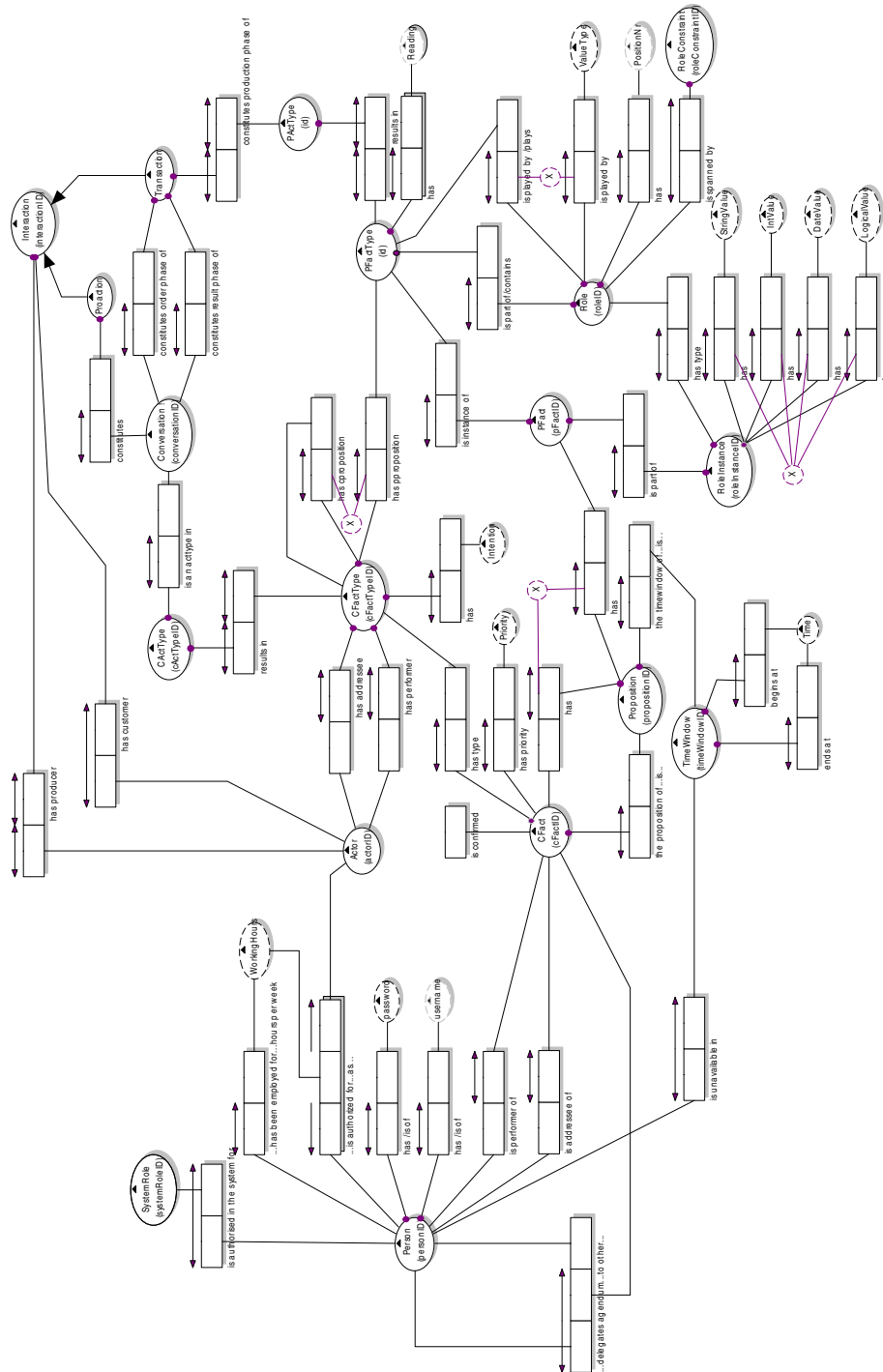
BIBLIOGRAFIE

Bijlage A

Diagrammen

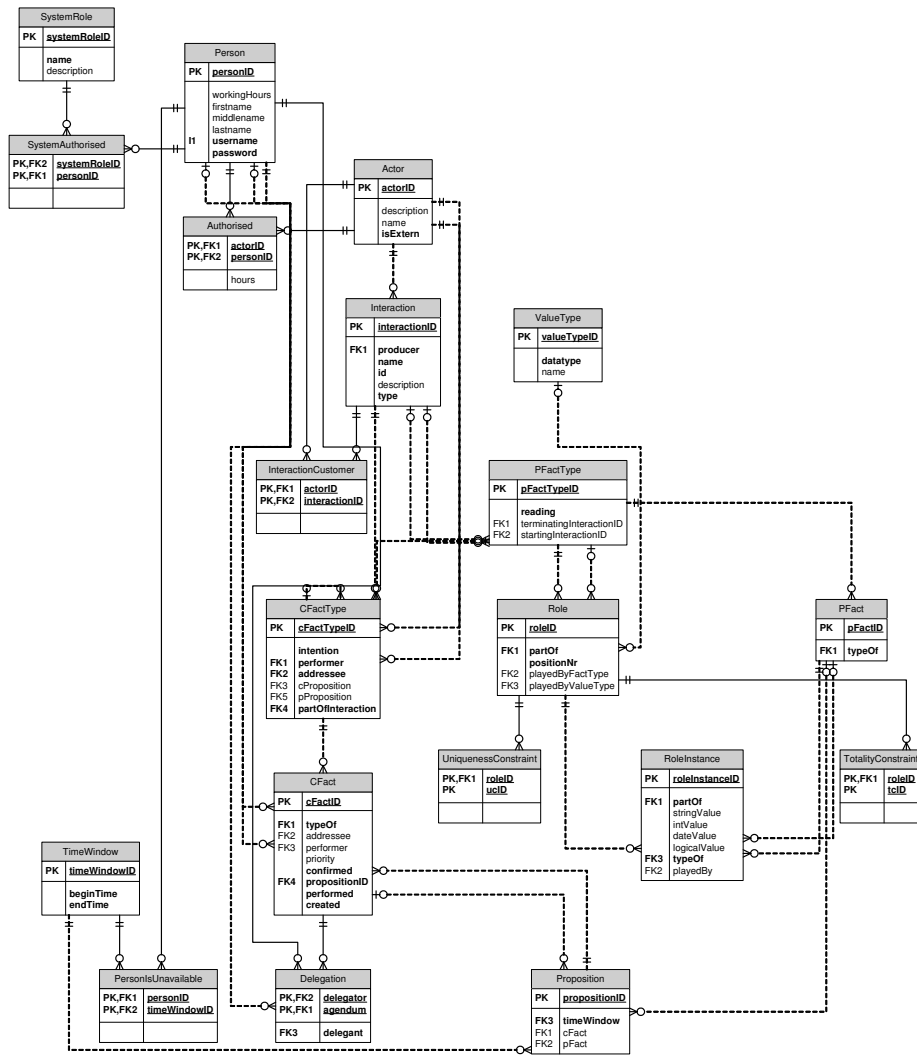


Figuur A.1: Feitenmodel van het bedrijfssysteem, bij hoofdstuk 3



Figuur A.2: Feitenmodel van het informatiesysteem, bij hoofdstuk 5

BIJLAGE A. DIAGRAMMEN



Figuur A.3: Database schema (in ERD) van het informatiesysteem, bij hoofdstuk 5

Bijlage B

Stored Procedures

```
CREATE PROCEDURE dbo.spPFact_Ins
    @typeOf smallint ,
    @pFactID smallint OUT
AS
INSERT INTO PFact (typeOf) VALUES (@typeOf)
SET @pFactID = SCOPE_IDENTITY()
GO

CREATE PROCEDURE dbo.spRoleInstance_Ins
    @typeOf smallint ,
    @partOf smallint ,
    @stringValue varchar(50) ,
    @logicalValue bit ,
    @intValue smallint ,
    @dateValue smalldatetime ,
    @playedBy smallint ,
    @roleInstanceID smallint OUT
AS
INSERT INTO RoleInstance(typeOf , partOf , stringValue ,
    logicalValue , intValue , dateValue , playedBy)
    VALUES (@typeOf , @partOf , @stringValue , @logicalValue ,
    @intValue , @dateValue , @playedBy)
SET @roleInstanceID = SCOPE_IDENTITY()
GO

CREATE PROCEDURE dbo.spCFact_Ins
    @priority smallint ,
    @confirmed bit ,
    @propositionID smallint ,
    @typeOf smallint ,
    @addressee smallint ,
    @performer smallint ,
    @performed bit ,
    @cFactID smallint OUT
AS
INSERT INTO CFact(priority , confirmed , propositionID , typeOf ,
    addressee , performer , performed)
```

BIJLAGE B. STORED PROCEDURES

```
VALUES (@priority , @confirmed , @propositionID , @typeOf
        , @addressee , @performer , @performed)
SET @cFactID = SCOPE_IDENTITY()
GO

CREATE PROCEDURE dbo.spProposition_Ins
    @timeWindow smallint ,
    @cFact smallint ,
    @pFact smallint ,
    @propositionID smallint OUT
AS
INSERT INTO Proposition(timeWindow , cFact , pFact)
VALUES (@timeWindow , @cFact , @pFact)
SET @propositionID = SCOPE_IDENTITY()
GO

CREATE PROCEDURE dbo.spTimeWindow_Ins
    @beginTime smalldatetime ,
    @endTime smalldatetime ,
    @timeWindowID smallint OUT
AS
INSERT INTO TimeWindow(beginTime , endTime)
VALUES (@beginTime , @endTime)
SET @timeWindowID = SCOPE_IDENTITY()
GO

CREATE PROCEDURE dbo.spClearDynamicData
AS
ALTER TABLE Proposition DISABLE TRIGGER Propositiondelete
ALTER TABLE CFact DISABLE TRIGGER CFactdelete
ALTER TABLE PFact DISABLE TRIGGER PFactdelete
ALTER TABLE TimeWindow DISABLE TRIGGER TimeWindowdelete
ALTER TABLE Proposition NOCHECK CONSTRAINT
    PFact_Proposition_FK1
ALTER TABLE Proposition NOCHECK CONSTRAINT
    CFact_Proposition_FK1
ALTER TABLE Proposition NOCHECK CONSTRAINT
    TimeWindow_Proposition_FK1

DELETE FROM RoleInstance;
DELETE FROM CFact;
DELETE FROM PFact;
DELETE FROM Proposition;
DELETE FROM TimeWindow;

ALTER TABLE Proposition ENABLE TRIGGER Propositiondelete
ALTER TABLE CFact ENABLE TRIGGER CFactdelete
ALTER TABLE PFact ENABLE TRIGGER PFactdelete
ALTER TABLE TimeWindow ENABLE TRIGGER TimeWindowdelete
ALTER TABLE Proposition CHECK CONSTRAINT PFact_Proposition_FK1
ALTER TABLE Proposition CHECK CONSTRAINT CFact_Proposition_FK1
ALTER TABLE Proposition CHECK CONSTRAINT
    TimeWindow_Proposition_FK1
GO
```

```

CREATE PROCEDURE dbo.spClearModelData
AS
DELETE FROM RoleInstance;
DELETE FROM PFact;
DELETE FROM ValueType;
DELETE FROM Role;
DELETE FROM TotalityConstraint;
DELETE FROM UniquenessConstraint;
DELETE FROM Authorised
DELETE FROM Actor;
DELETE FROM CFactType;
DELETE FROM Interaction;
DELETE FROM InteractionCustomer;
DELETE FROM PFactType;
GO

CREATE PROCEDURE dbo.spGetOutstandingConfirmations
    @personID smallint
AS
SELECT * FROM CFact
WHERE performed = 1 AND confirmed = 0 AND addressee = @personID
GO

CREATE procedure dbo.spGetAgenda
    @personID smallint
AS
SELECT CFact.* FROM CFact JOIN CFactType on CFact.typeOf =
    CFactTypeID WHERE (
        cfactID IN (
            SELECT MAX(CFact.cFactID)
            FROM (CFact JOIN CFactType ON CFact.typeOf =
                CFactType.cFactTypeID) JOIN Person ON CFact.
                addressee = Person.personID
            WHERE performed = 1 AND confirmed = 1
            GROUP BY propositionID
        )
        AND
        (
            ((CFact.addressee = @personID) AND (intention
                = 'rq' OR intention = 'st' OR intention = '
                dc' OR intention = 'rj' OR intention = 'av'))
            OR
            ((CFact.performer = @personID) AND (intention
                = 'pm' OR intention = 'ut'))
        )
        AND CFact.propositionID NOT IN (
            SELECT propositionID FROM CFact WHERE confirmed
                = 0
        )
    )
)

```

BIJLAGE B. STORED PROCEDURES

```
GO

CREATE PROCEDURE dbo.spGetCommunicationNotAgenda
    @personID smallint
AS
SELECT CFact.* FROM ( CFact JOIN CFactType ON CFact.typeOf =
    cFactTypeID)
WHERE cFactID IN
(
    SELECT MAX(CFact.cFactID)
    FROM ( CFact JOIN CFactType ON CFact.typeOf = CFactType.
        cFactTypeID)
    JOIN Person ON CFact.addressee = Person.personID
    WHERE performed = 1 AND confirmed = 1
    GROUP BY propositionID
)
AND
(
    (CFact.addressee = @personID)
    AND
    (intention = 'pm' OR intention = 'qt' OR intention = '
        sp' OR intention = 'ac' OR intention = 'ut')
)
GO

CREATE PROCEDURE dbo.spGetObligatoryPFactTypes
    @beginID smallint
AS
    SELECT PFactType.pFactTypeID
    FROM Role JOIN PFactType ON Role.partOf = PFactTypeID
    WHERE playedByFactType = @beginID
    AND Role.roleID IN (
        SELECT roleID FROM TotalityConstraint
    )
GO

CREATE PROCEDURE dbo.spGetAllRelationsPlayedIn
    @beginID smallint
AS
    SELECT PFactType.*
    FROM Role JOIN PFactType ON Role.partOf = PFactTypeID
    WHERE playedByFactType = @beginID
GO

CREATE PROCEDURE dbo.spGetObligatoryPFacts
    @beginID smallint
AS
SELECT *
FROM PFact
WHERE pFactID in (
    SELECT DISTINCT partOf
```

```

        FROM RoleInstance JOIN PFact ON playedBy = @beginID
        WHERE RoleInstance.typeOf IN (
            SELECT roleID FROM TotalityConstraint
        )
    )
GO

CREATE PROCEDURE dbo.spGetActivePFacts
    @typeID smallint
AS
SELECT pFactID
FROM
    CFact JOIN CFactType ON CFact.typeOf = cFactType.cFactTypeID
    JOIN Proposition ON CFact.propositionID = Proposition.
        propositionID
    JOIN PFact ON Proposition.PFact = PFact.pfactID join pfacttype
        ON PFact.typeOf = PFacttype.pFactTypeID
WHERE intention = 'ac' AND pfacttype.pFactTypeID = @typeID

UNION
SELECT pFactID FROM
    PFact JOIN PFacttype ON PFact.typeOf = PFactType.PFactTypeID
WHERE startingInteractionID IS NULL AND pFactTypeID = @typeID
GO

CREATE PROCEDURE dbo.spPerson_Ins
    @workingHours smallint ,
    @firstname varchar(20) ,
    @middlename varchar(20) ,
    @lastname varchar(30) ,
    @username varchar(10) ,
    @password varchar(10) ,
    @personID smallint OUT
AS
INSERT INTO Person(workingHours , firstname , middlename ,
    lastname , username , password)
    VALUES ( @workingHours , @firstname , @middlename ,
        @lastname , @username , @password)
SET @personID = SCOPE_IDENTITY()
GO

```

BIJLAGE B. STORED PROCEDURES

Bijlage C

Introductie tot DEMO

DEMO is een methodiek voor het modelleren en analyseren van bedrijfsprocessen. Het doel van het opstellen van die modellen is te begrijpen en te verklaren hoe bedrijfsprocessen in elkaar zitten. Een methodiek baseert zich in principe op een bepaalde denkwijze over de fenomenen die het modelleert. De manier waarop je een fenomeen modelleert is afhankelijk van die denkwijze. De denkwijze die de basis vormt voor de methodiek DEMO heet de CAP-theorie (oa. in [13] en [15]), hetgeen een acroniem is van Coördinatie, Actoren en Productie.

Volgens DEMO is het doel van een organisatie het coördineren van de inspanningen van een aantal mensen die aan een gezamenlijke taak werken. DEMO beschouwt een organisatie daarom als een netwerk van verplichtingen; in een organisatie communiceren mensen waarbij ze verplichtingen aangaan om de gewenste veranderingen in de wereld tot stand te brengen. DEMO gaat uit van het beginsel dat het beschrijven van een organisatie teruggebracht kan worden tot een bepaalde essentie. DEMO brengt daarom een scheiding aan tussen het essentiële, abstracte, model van een organisatie en de manier waarop die essentie in de werkelijke situatie is ingevuld. Als er bijvoorbeeld een pizzeria bestaat die pizza's levert, waarin drie medewerkers werken: een kok, een receptionist en een bezorger, dan is van de pizzeria een bedrijfsmodel te maken. DEMO gaat nu uit van het standpunt dat als er –in een tweede situatie– twee bezorgers bij komen, zodat de jongens nu met z'n drieën zijn, dit natuurlijk wel een verandering is op de werkvloer, maar dat er in het essentiële model van de organisatie niets veranderd is. DEMO probeert nu juist dat essentiële model te beschrijven en zich niet te richten op de operationele invulling van dat essentiële model. Om de essentie van de organisatie te vatten, moeten de verplichtingen die de mensen aangaan worden beschouwd, en daarvoor moet hun onderlinge communicatie worden beschouwd. De communicatie wordt daarbij vanuit het Language/ Action perspectief benaderd, waarbij volgens Auramäki [7] elementen uit het werk van zowel Austin, Searle als Habermas gebruikt worden.

C.1 Language/ Action perspectief

In een klassieke linguïstische benadering van communicatie wordt taal gezien als een manier om informatie over te brengen, om de wereld te beschrijven. Als de bakker tegen de klant zegt “dit brood kost één euro per stuk” dan wordt dat gezien als het overbrengen van informatie over een bepaalde stand van zaken in de wereld. Het Language/ Action perspectief daarentegen gaat ervan uit dat taal niet alleen een middel is

om informatie over te brengen, maar ook een performatief aspect bezit. Taal is in die opvatting niet slechts een manier om informatie over te brengen, maar ook een manier om acties te coördineren. Taal beschrijft niet slechts een toestand maar is er ook op gericht om een toestandsverandering tot stand te brengen.

Het Language/ Action perspectief werd door Austin geïntroduceerd in zijn boek *How to Do Things with Words* (1962). Hij zet hierin zijn theorie van *speech-acts* uiteen, waarbij hij zegt dat onze opvatting over de functie van taal moet worden uitgebreid van “slechts een manier om dingen te beschrijven” naar “een manier om dingen te doen”. De uitbreidingen die in de loop van de tijd zijn gedaan op het oorspronkelijke werk van Austin ertoe hebben geleid dat we nu, volgens Auramäki [7], vier speech-act theorieën kunnen onderscheiden in achtereenvolgens het werk van Austin, Searle, Habermas en Ballmer & Brennenstuhl. Deze vier theorieën onderscheiden zich door verschillen in de benadering van onder andere de classificatie van de speech-acts, het belang van de context en het concept van commitments, en de manier waarop deze tot stand komen.

Volgens Dignum et al. [17] is het Language/ Action perspectief voor het eerst op het terrein van de informatiesystemen gebracht door Flores en Ludlow, die voortbouwden op de interpretatie van Searle. Van Reijswoud [25] geeft aan dat hun werk een inspiratiebron is geweest voor de ontwikkeling van DEMO. DEMO is daarnaast voortgekomen uit een gemis in de toenmalige informatiekundige methoden: informatiesystemen worden namelijk ingezet ter ondersteuning van een organisatie, of een bepaald proces binnen die organisatie, hetgeen betekent dat er dus een bepaalde afstemming moet plaatsvinden tussen het ontwerp van het informatiesysteem en de structuur en werking van de organisatie. Deze afstemming is in de loop van de tijd dusdanig belangrijk geworden dat aan ingenieurs niet meer wordt gevraagd slechts informatiesystemen te ontwerpen, maar zich ook bezig te houden met het (her)ontwerpen van de bedrijfsprocessen. Maar dit betekent dat de methoden die hen ter beschikking staan dit wel mogelijk moeten maken, en een begrip moeten bieden van zowel informatiesystemen als organisaties. Dit was in de toen beschikbare ontwerpmethoden niet het geval [12]. DEMO tracht dat begrip wel te bieden, en een brug te slaan tussen organisaties en informatiesystemen;

C.2 CAP-theorie

CAP onderschrijft, in navolging van het eigen acroniem, drie basisbegrippen in een organisatie: ten eerste is er in elke organisatie sprake van een bepaalde productie: wat maakt de organisatie, wat levert ze aan de buitenwereld. Ten tweede zijn er binnen en buiten een organisatie een aantal partijen, bijvoorbeeld personen of afdelingen, die een rol spelen in de productie: zij zijn verantwoordelijk voor een bepaald deel van de productie; deze partijen noemen we actoren. Ten derde vindt er coördinatie plaats tussen deze verschillende partijen waardoor ze de productie tot stand brengen.

C.2.1 Beginselen

Volgens Dietz [13] is een organisatie een systeem dat bestaat uit individuen die twee soorten acties uitvoeren: productieve¹ en coördinatieve acties. De productieve acties

¹Het woord *productief* moet niet worden opgevat in de betekenis *veel opbrengend of voortbrengend, vruchtbaar of winstgevend*. Productieve acties in CAP zijn alle acties die deel uitmaken van de productie van een organisatie, in tegenstelling tot coördinatieve acties, die de productie coördineren.

kunnen zowel materieel als immaterieel zijn; zij dragen per definitie bij aan het bedrijfsdoel (missie) van de organisatie. Het resultaat van een productieve actie is een productief feit. Deze productieve feiten kunnen dus materieel van aard zijn, zoals bij een fietsenfabriek, maar kunnen ook bestaan uit een bepaalde dienst, zoals een beleggingsbank die andermans geld investeert of een rechtbank die juridische oordelen velt. Twee productieve feiten die bijvoorbeeld in een bibliotheek tot stand kunnen komen zijn:

Lidmaatschap L is ingegaan.

Boek B is uitgeleend aan Lid L op datum D.

De productie van een organisatie wordt gevormd uit de verzameling van alle productieve feiten die in de organisatie worden geproduceerd. In het dagelijks leven zeggen we niet vaak dat een rechtbank producten levert, maar volgens de CAP-theorie is dat wel het geval.

De tweede type acties die de actoren in de organisatie uitvoeren, zijn coördinatieve acties. Dit zijn feitelijk communicatieve acties, in de lijn van de speech-act theorie. De term *coördinatieve actie* heeft betrekking op de rol die deze communicatie speelt, namelijk elkaars gedrag coördineren [13]. Deze coördinatieve acties bestaan er dus uit dat de actoren afspraken met elkaar maken, en dus verplichtingen aangaan, over het tot stand brengen van de productiefactoren. Het resultaat van deze coördinatieve acties, die dus bedoeld zijn om de productie te sturen, zijn coördinatieve feiten.

Als bijvoorbeeld in een bibliotheek iemand lid wil worden dan vraagt hij iets als: “kan ik hier lid worden”. In de CAP-theorie is dit een speech-act die bestaat uit achtereenvolgens een performer, intentie, addressee en een propositie. De ene partij gaat dus een coördinatieve actie aan met de andere waarvan het resultaat het volgende coördinatieve feit is:

John verzoekt Mary: lidmaatschap 387 is ingegaan op 01/06/2002.

De performer (John) is degene die iets zegt tegen de addressee (Mary). De propositie is een bepaalde stand van zaken waarover hij iets zegt (het lidmaatschap is ingegaan) en de intentie is hetgeen hij tot stand wil brengen met zijn communicatieve actie. De propositie hoeft geen stand van zaken te zijn die reeds bestaat, het kan ook een toekomstige stand van zaken zijn. In dit geval verzoekt John aan Mary dat de stand van zaken waarbij zijn lidmaatschap is ingegaan bestaat, en dus dat Mary John tot lid verklaart. Mary zou hierop bijvoorbeeld kunnen antwoorden dat de bibliotheek helaas geen nieuwe leden meer aanneemt. In de CAP-theorie is ook dit een speech-act die er als volgt uitziet:

Mary weigert John: lidmaatschap 387 is ingegaan op 01/06/2002.

Hierbij is Mary de performer (zij zegt iets) en John de addressee (tegen wie Mary iets zegt). De propositie is dezelfde gebleven, zij praten beiden over dezelfde stand van zaken. En waar John net vroeg om deze propositie tot stand te brengen, weigert Mary dat nu te doen. De precieze definitie van de structuur van een coördinatieve actie is gegeven in [13].

De CAP-theorie maakt ten derde gebruik van het begrip actor om de focus weg te nemen van de personen in de organisatie, en deze te richten op de rollen die binnen een organisatie gespeeld worden. Het feit dat personen dingen doen in een organisatie komt doordat er een bepaalde operationele invulling is gegeven aan de rollen die gespeeld moeten worden. Aangezien CAP zich erop richt om de essentie van de organisatie te beschouwen, en niet een bepaalde invulling van die essentie, is het begrip actor

geïntroduceerd. Een actor is gedefinieerd [13] als “een bepaalde hoeveelheid autoriteit (en daarmee een bepaalde verantwoordelijkheid en competentie) om precies één type productiefacten tot stand te brengen”. Een actor is dus een bepaalde rol binnen een organisatie die verantwoordelijk is voor het tot stand komen van één type productiefacten. Een actor is een abstractie van personen of afdelingen binnen de organisatie; een actor kan door meerdere personen vervuld worden en één persoon kan meerdere actoren vervullen.

Actoren hebben een levenscyclus waarin ze steeds reageren op bepaalde coördinatieve facten en aan de hand van een actieregel één of meer actie(s) ondernemen. Als in het genoemde voorbeeld van de bibliotheek John aan Mary verzoekt om lid te worden, dan brengt hij daarmee een coördinatief feit tot stand. Dat feit geldt als agendum voor Mary, dat wil zeggen dat het voor Mary een reden is om tot actie over te gaan. Zij reageert namelijk met een of meer actie(s), zoals haar actieregel dat voorschrijft. Dit kan een coördinatieve actie zijn, bijvoorbeeld dat zij afwijst dat hij lid kan worden, maar ook een productieve actie, bijvoorbeeld dat zij een lidmaatschap aanmaakt voor hem.

Het is belangrijk het verband in te zien tussen de coördinatieve acties en de productieve acties van actoren. Volgens het Language/ Action perspectief kan een uiting van taal worden gezien als het doen van een actie. Maar de CAP-theorie gaat een stap verder: een productieve actie is altijd het gevolg van een –eerder aangegane– verplichting, die voortkomt uit een coördinatieve actie. Volgens de definitie die CAP gebruikt bestaat een organisatie altijd uit een gemeenschappelijk doel, en werken de individuen daarbinnen aan een gezamenlijke taak. Voor het bereiken van die taak maken ze met elkaar afspraken en de verplichtingen die uit die afspraken volgen leiden ertoe dat de actoren productieve acties gaan uitvoeren. Er is dus nooit sprake van een productieve actie die niet volgt uit een coördinatieve actie, er is dus ook nooit een productieve actie die niet bijdraagt aan het gemeenschappelijke doel.

C.2.2 Hiërarchische bouwstenen: atomen, moleculen en materie

Voortbouwend op de bovenstaande benadering introduceert Dietz in [13] een kader waarin, vanuit een Language/ Action perspectief, een hiërarchische opbouw wordt aangebracht in organisaties. Hij spreekt daarin over atomen, moleculen en materie van organisaties.

Atomen Dietz beschouwt de coördinatieve acties als één van de twee bouwstenen van een model van een organisatie. Dit komt voort uit de gedachte dat communicatie als actie opgevat kan worden, omdat coördinatie (ofwel communicatie) leidt tot verplichtingen, en verplichtingen slechts ingelost kunnen worden door productieve acties. Dit is per definitie zo, aangezien de coördinatie ten allen tijde het bereiken van het gezamenlijke doel, de gemeenschappelijke taak, betreft en de coördinatie dus altijd leidt tot een afspraak en een verplichting om dichterbij het doel te komen. Zo leiden coördinatieve acties dus altijd óf tot andere coördinatieve acties, óf tot productieve acties. Daarnaast beschouwt Dietz de actieregels als atomaire delen van een bedrijfsproces. Dit komt omdat de actieregels bepalen hoe elke actor reageert op bepaalde coördinatieve facten. De reactie van de actor bestaat uit één (of meerdere) coördinatieve actie(s) en eventueel een productieve actie. Dus daar waar de coördinatieve acties de oorzaak zijn van alle productieve acties, zijn de actieregels de lijm die de verschillende coördinatieve en productieve acties aan elkaar relateren.

Moleculen Op het tweede hiërarchische niveau (de ‘moleculen’ van een organisatie) onderkent Dietz twee concepten: de proactie en de transactie. Dit zijn beiden conversaties. Een conversatie is “een eindige sequentie van coördinatieve acties die betrekking hebben op eenzelfde propositie en door twee actoren afwisselend worden uitgevoerd” [13]. Het voorgenoemde voorbeeld, waarin John eerst een coördinatieve actie uitvoerde tegen Mary, waarin hij verzocht lid te worden, en waarin later Mary hem dat weigerde, is een conversatie. De twee coördinatieve acties hebben namelijk beiden betrekking op dezelfde propositie, namelijk ‘Lidmaatschap 387 is ingegaan op 01/06/2002’, waarbij John deze propositie verzocht en Mary deze propositie weigerde.

Het concept conversatie kan gemodelleerd worden als een petri-net. Er is namelijk sprake van (coördinatieve) acties die leiden tot (coördinatieve) feiten, hetgeen kan worden opgevat als een petri-net waarin de acties als transities worden gemodelleerd en de resulterende feiten als toestanden. Er bestaan wel een aantal verschillen ten opzichte van klassieke of gekleurde petri-netten hetgeen reden heeft gegeven deze modellering van conversaties CAP-net te noemen. Een belangrijke eigenschap van CAP-net is dat (in tegenstelling tot klassieke petri-netten) een transitie precies één toestand als uitgang heeft, aangezien elke (coördinatieve of productieve) actie precies één (coördinatief of productief) feit tot gevolg heeft. Een conversatie is dus een opeenvolging van coördinatieve acties (en dus coördinatieve feiten) die door twee actoren worden uitgevoerd.

Nadat John verzoekt om lid te worden van de bibliotheek kan Mary ook reageren met een belofte in plaats van een weigering. Dat is eigenlijk de normale gang van zaken aangezien de bibliotheek in het algemeen juist graag nieuwe leden krijgt. Ook dit zou dan een conversatie zijn, het verzoeken om een bepaalde propositie door één actor en het beloven van die propositie door de andere actor. Deze conversatie komt zo vaak voor dat er een naam aan gegeven is, namelijk een opdrachtconversatie, of O-conversatie. Omdat de actor die heeft beloofd de propositie tot stand te brengen de verantwoordelijkheid heeft deze ook werkelijk tot stand te brengen, zal deze actor nu ofwel zelf een productieve actie uitvoeren waarmee de propositie tot stand wordt gebracht, of een andere actor verzoeken dit te doen. Dus nadat Mary heeft beloofd John lid te maken, zal ze zelf het lidmaatschap voor hem aanmaken (ze zal bijvoorbeeld de gegevens van John in de computer invoeren en een ledenpas voor hem maken) of een collega vragen dit te doen.

Vervolgens vertelt ze John dat het productieve feit, de propositie waar hij om verzocht, tot stand is gebracht: ze overhandigt hem bijvoorbeeld zijn nieuwe pas en zegt tegen hem dat hij zichzelf vanaf dat moment kan beschouwen als lid van de bibliotheek met alle rechten en plichten die daarbij horen. Maar het kan op dat moment ook voorkomen dat John ziet dat op de pas zijn naam verkeerd is gespeld en hij tegen Mary zegt dat ze dit eerst moet verbeteren. Hij zegt dan eigenlijk dat, hoewel zij verklaart dat de propositie ‘Lidmaatschap 387 is ingegaan op 01/06/2002’ tot stand is gebracht, hij het daar niet mee eens is, aangezien zij daarbij een fout heeft gemaakt. Er vindt dus altijd nog een conversatie plaats, nadat het productieve feit tot stand is gebracht, waarbij de actor die daarvoor verantwoordelijk was verklaart dat de propositie tot stand is gebracht en de andere actor dit al dan niet aanvaardt. Het feit dat John lid is van de bibliotheek komt dus volgens de CAP-theorie pas werkelijk tot stand als John het hiermee eens is, als John bevestigt dat Mary zijn oorspronkelijke verzoek om hem lid te maken, correct heeft ingewilligd. Deze conversatie (die dus bestaat uit twee coördinatieve acties, het verklaren en vervolgens aanvaarden van een bepaalde propositie) heeft ook een eigen naam: een resultaatconversatie, of R-conversatie.

Dit patroon waarbij een O-conversatie gevolgd wordt door een productieve actie waarna een R-conversatie plaatsvindt heet in de CAP-theorie een transactie. De

O-conversatie vormt de opdrachtfase (O-fase) van de transactie, de productieve actie vormt de executiefase (E-fase) van de transactie en de R-conversatie vormt de resultaat-fase (R-fase) van de transactie. In de O-fase wordt (door middel van een O-conversatie) een opdracht gegeven, in de productieve fase komt door de productieve actie het gewenste resultaat tot stand, waarna in de R-fase wordt bediscussieerd of het resultaat voldoet aan de opdracht. Slechts indien, in de R-conversatie, het geproduceerde resultaat –door de actor die de transactie startte– wordt aanvaard, is het productieve feit werkelijk tot stand gekomen.

Zowel in de O-conversatie als in de R-conversatie hebben beide actoren de mogelijkheid om te discussiëren over de propositie, zoals we reeds zagen in het voorbeeld van de bibliotheek waarbij John een discussie kon beginnen over de verklaring van Mary dat zijn verzochte propositie tot stand was gebracht. Maar ook in de O-conversatie is er ruimte voor discussie: als bijvoorbeeld de propositie het lenen betreft van een bepaald boek, en dat boek is toevallig uitgeleend, dan heeft Mary ten eerste natuurlijk de mogelijkheid om de propositie die John verzoekt af te wijzen. Maar ze kan ook met John in discussie gaan of hij zijn propositie niet een beetje wil aanpassen, en bijvoorbeeld een ander boek van dezelfde auteur wil lenen, waarna ze de vernieuwde propositie wél belooft tot stand te brengen.

De mogelijke gronden voor deze discussies zijn de drie geldigheidsaanspraken van Habermas. Habermas stelt (volgens Verheggen [27]) dat het doel van communicatie het coördineren van acties is. Deze coördinatie vindt plaats door te zoeken naar een vorm van wederzijds begrip. De ene partij probeert de andere partij over te halen zijn wens uit te voeren of eraan mee te werken. Het besluit van de andere partij om te helpen het gewenste tot stand te brengen is volgens Habermas gebaseerd op het feit dat deze één van de drie zogenaamde geldigheidsaanspraken (*waarheid, juistheid of waarachtigheid*) erkennen. Bij elke coördinatieve actie claimt degene die de actie doet namelijk een bepaalde validiteit, of echtheid, van zijn uitspraak. Kort gezegd kan de spreker stellen dat zijn uitspraak waar is, dat wil zeggen in overeenstemming met de feiten (aanspraak op de waarheid), dat zijn uitspraak goed is volgens bepaalde normen (aanspraak op juistheid) of dat zijn uitspraak overeenkomt met zijn gedachten, dus dat hij eerlijk is (aanspraak op waarachtigheid). Op het moment dat één van de partijen de claim niet onderkent vindt er een onderhandeling plaats of wordt de samenwerking beëindigd. Juist deze geldigheidsaanspraken van Habermas en het niet erkennen daarvan door één van de actoren, zijn volgens de CAP-theorie in een conversatie reden tot discussie of afwijzing, zoals in het voorbeeld van de bibliotheek.

Als tweede concept op het moleculaire niveau onderscheidt Dietz een ander patroon van coördinatieve acties. Een actor kan namelijk ook proberen een andere actor over te halen om een bepaalde transactie in gang te zetten, bijvoorbeeld als Jane tegen John zegt dat hij eigenlijk lid zou moeten worden bij de bibliotheek. Hierbij probeert zij hem dus ertoe te brengen om de transactie in gang te zetten om dat lidmaatschap te verkrijgen. Feitelijk adviseert zij hem niet om de hele transactie in gang te zetten, maar slechts om een bepaald coördinatief feit tot stand te brengen. De coördinatieve actie die zij hiervoor uitvoert is:

Jane adviseert John (John verzoekt Mary: lidmaatschap 387 is ingegaan op 01/06/2002).

John kan op deze aansporing van Mary reageren door of te weigeren of te zeggen dat hij dit zal ondernemen, waarna hij de transactie *aanvragen lidmaatschap* zal starten. Het patroon dat deze twee coördinatieve acties samen vormen heet een proactie. Een

proactie is dus een conversatie waarin de ene actor de andere actor aanzet om een bepaald coördinatief feit tot stand te brengen.

Daarnaast behoren ook de actoren tot de moleculen van de organisatie. De actoren zijn het dynamische aspect, die ervoor zorgen dat er coördinatie en dus productie bestaat. Een actor bestaat uit de verzameling van zijn actieregels, oftewel een verzameling coördinatieve feiten die voor hem aanleiding zijn om een actie te ondernemen samen met de definitie van die acties. De mogelijke interactie die een actor kan hebben met andere actoren bestaat uit transacties en proacties.

Materie Het derde niveau bestaat uit bedrijfsprocessen. Het concept bedrijfsproces is een bundeling van een aantal transacties en proacties die met elkaar samenhangen. Als bijvoorbeeld voor het succesvol slagen van de transactie *aanvragen lidmaatschap* eerst de transactie *lidmaatschap betalen* succesvol moet worden afgerond, dan zijn deze transacties dus met elkaar verbonden. Deze zouden samen het bedrijfsproces *lidmaatschap verkrijgen* kunnen vormen. Zo kan dus uit moleculen, en dus uit atomen, de materie van een organisatie worden samengesteld

C.3 DEMO modelleertechniek

DEMO benadert een organisatie als een discreet dynamisch systeem. Een organisatie is ten eerste een *systeem* omdat er een grens te trekken is tussen een kern die bestaat uit elkaar beïnvloedende elementen die activiteiten ontplooiën en de buitenwereld. Het is daarnaast een *discreet dynamisch* systeem omdat de activiteiten die in het systeem plaatsvinden niet-continue (dus discrete) toestandsveranderingen tot gevolg hebben. Dat discrete dynamische systeem is vervolgens volledig te modelleren door vier aspecten ervan te bekijken: de constructie, de toestandswereld, het procesverloop en de operationele werking. De constructie van een organisatie geeft aan welke elementen er bestaan en in welke relatie die tot elkaar staan. Ten tweede hebben de elementen een beperkt aantal toestanden waarin ze zich kunnen bevinden die samen de toestandswereld van de organisatie vormen. Ten derde heeft elke organisatie een bepaald procesverloop waarin de mogelijke overgangen tussen de verschillende toestanden staan. Ten vierde kent elke organisatie operationele regels die voorschrijven welke toestandsovergangen in een individueel geval plaatsvinden. Deze vier aspecten komen terug in de vier modellen die met DEMO te maken zijn van een organisatie [14]: het constructie-, toestand-, proces- en operationeel model, die samen de gehele afbeelding vormen die DEMO maakt van een organisatie. Deze vier modellen gaan uit van de CAP-theorie.

Het constructiemodel geeft aan uit welke elementen het systeem bestaat en in welke relatie zij tot elkaar staan. De elementen waaruit het systeem in DEMO bestaat zijn alle actoren; de relaties die tussen de elementen bestaan zijn de mogelijke interacties (zowel transacties als proacties) die zij met elkaar aangaan. Dat zijn nu juist de drie beginselen van CAP: actoren produceren feiten, en coördineren daarbij onderling hun activiteiten door middel van transacties en proacties.

Het toestandsmodel beschrijft alle mogelijke soorten feiten die in het systeem kunnen voorkomen, de toestandswereld. De toestanden worden bereikt door toestands-overgangen, die bestaan uit hetzij coördinatieve acties, hetzij productieve acties. De ene heeft een totstandkoming van een nieuw feit in de coördinatiewereld tot gevolg, de ander een totstandkoming van een nieuw feit in de productiewereld. De toestandswereld bestaat dus uit de verzameling van alle mogelijke coördinatieve en productieve feiten.

Het procesmodel definieert het procesverloop en geeft dus aan welke toestandsovergangen er mogelijk zijn tussen de, in het toestandsmodel gedefinieerde, toestanden. Aangezien alle toestandsovergangen bestaan uit coördinatieve en productieve acties, en deze acties altijd plaatsvinden binnen een interactie, bestaat het procesmodel uit de gedetailleerde uitwerking van alle interacties. In deze uitwerking kan worden gemodelleerd welke coördinatieve en productieve acties gedaan worden tijdens bepaalde interacties en of deze eventueel bepaalde causale of conditionele afhankelijkheden kennen.

Het operationele model definieert tenslotte de werking van de actoren uit CAP. Elke actor handelt namelijk volgens bepaalde actieregels, die aangeven wanneer hij iets doet. Alle mogelijke toestandsovergangen waaruit hij kan kiezen staan weergegeven in het procesmodel. Het operationele model definieert voor elke toestand de actie die de actor uitvoert, en dus welke van de mogelijke toestandsovergangen de actor realiseert.

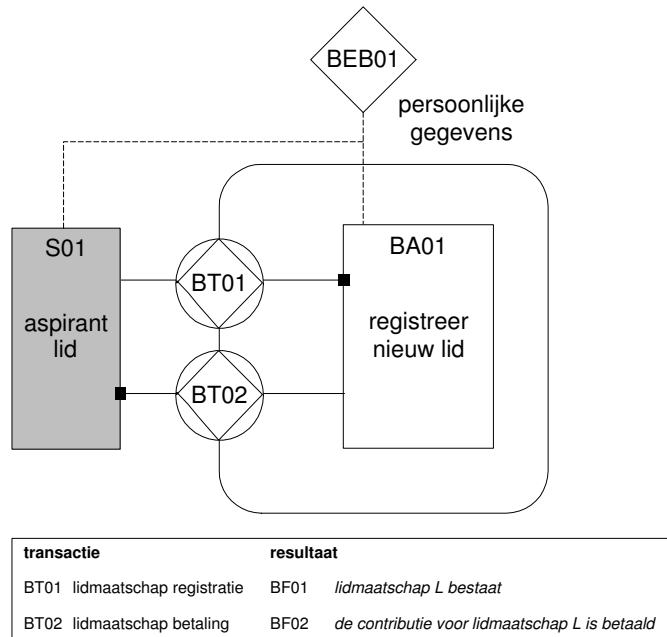
DEMO diagrammen De DEMO-modellen zijn een abstracte specificatie van een organisatie en bestaan uit een aantal diagrammen en aanvullende tabellen [14]. Voor ons onderzoek zijn het constructiemodel en het procesmodel relevant. Het constructiemodel wordt weergegeven in het coördinatie-diagram, waarin de interacties en interstricties staan die in het systeem voorkomen, en de transactie-conversatie tabel waarin per transactie het productieve feit genoemd staat dat tot stand komt. Het procesmodel wordt weergegeven in het procesfasediagram en het processtapdiagram. In het processtapdiagram staat per interactie welke coördinatieve en productieve acties plaatsvinden (en welke actor deze uitvoert). In het procesfasediagram staan niet meer de coördinatieve en productieve acties genoemd, maar slechts de fasen van de transactie. Het feitmodel wordt weergegeven in het feitendiagram, waarin de feittypen staan die in het model voorkomen, de relaties tussen de verschillende feittypen en het verband tussen een transactie en het feittype dat door die transactie tot stand wordt gebracht.

C.3.1 Coördinatie-diagram

In het coördinatie-diagram is te zien welke coördinatie er optreedt tussen de verschillende actoren. Hiervoor is allereerst een systeemgrens aangegeven waarmee wordt bepaald welke actoren wel en niet tot de organisatie worden gerekend. Daarnaast is aangegeven welke actoren er bestaan waarbij actoren die buiten het systeem vallen grijs worden gekleurd. Tenslotte is de coördinatie aangegeven die plaatsvindt tussen de verschillende actoren. Deze coördinatie kan bestaan uit het doen van transacties en proacties.

Een actor heeft tijdens het uitvoeren van zijn actieregel soms kennis nodig van bepaalde feiten. De actor *registreer nieuw lid* uit het voorbeeld van de bibliotheek, moet voor het uitvoeren van zijn productieve actie (het verschaffen van een lidmaatschap) bijvoorbeeld de leeftijd van het aspirant-lid kennen om de hoogte van de contributie te kunnen bepalen. Deze informatie-overdracht maakt geen deel uit van de transactie, in de propositie (van de tijdens de transactie uitgevoerde coördinatieve acties) immers staan deze reeds bestaande feiten niet opgenomen. De actor kan deze feiten te weten komen door ze op te zoeken in een feitenbank. In dit voorbeeld bestaat de feitenbank simpelweg uit het aspirant-lid: deze zal zelf zijn adresgegevens kunnen verschaffen. Maar in andere gevallen kan bijvoorbeeld een rijbewijs dienen als feitenbank, of een ander bedrijf. Om aan te geven dat een actor informatie nodig heeft tijdens transactie wordt in het diagram een gestippelde getekend naar een feitenbank. Een transactie kan

ook als feitenbank dienen, daarin kan de actor bijvoorbeeld opzoeken of een bepaalde transactie succesvol is afgerond. Ook kan een actor in een externe feitenbank informatie opzoeken, deze wordt weergegeven met een ruit. In figuur C.1 is het voorbeeld van de bibliotheek gemodelleerd.



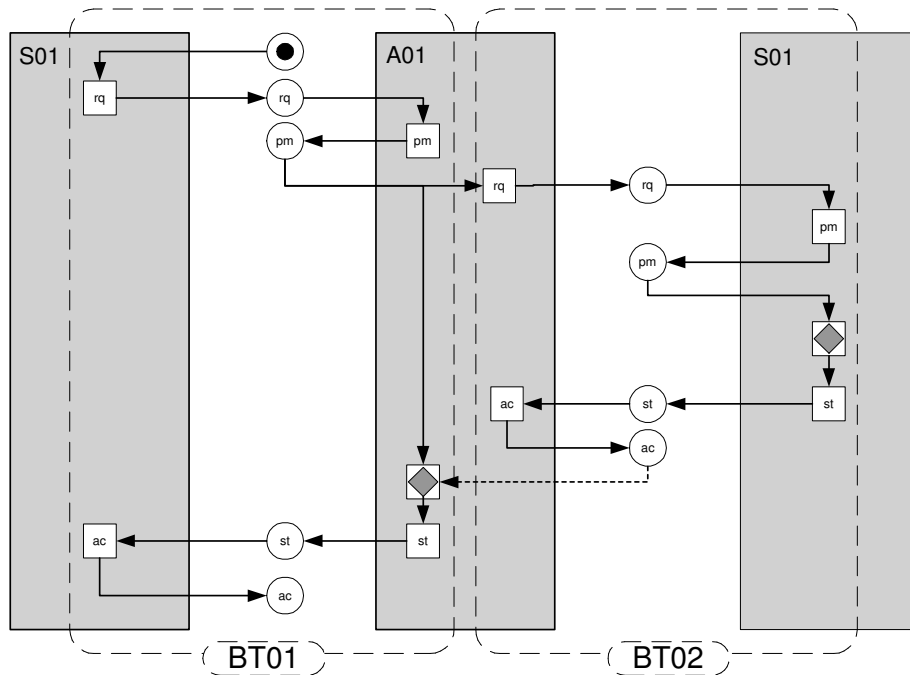
Figuur C.1: Coördinatiediagram bibliotheek, uit [14]

We zien hierin de bibliotheek gemodelleerd, met twee actoren (rechthoek), één externe feitenbank (ruit) en twee transacties (ruit in cirkel). In het voorbeeld vervult John de actor *aspirant-lid* (S01) en Mary de actor *registreer nieuw lid* (A01). De systeemgrens is zodanig gekozen dat de actor S01 buiten de systeemgrens valt, nieuwe leden (actor S01) maken dus geen deel uit van de gemodelleerde organisatie. Tussen de twee actoren vinden twee transacties plaats: BT01 en BT02. De productieve feiten die in deze transacties tot stand komen staan genoemd in de transactie-conversatie tabel eronder. Aan het zwarte blokje is te zien welke actor de uitvoerder is van deze transactie: de actor zonder blokje verzoekt aan de andere actor het productieve feit tot stand te brengen dat genoemd is. Ook is de interstrictie die plaatsvindt te zien: beide actoren inspecteren de externe feitenbank (BEB01) waarin persoonlijke gegevens staan. De actor S01 zet daar feiten in en de actor A01 zoekt deze feiten vervolgens op.

C.3.2 Processtapdiagram

De transacties die in het coördinatiediagram zijn gemodelleerd bestaan uit coördinatieve (en productieve) acties van de actoren. Het verloop van al deze acties en hun onderlinge afhankelijkheden kunnen weergegeven worden in zowel het processtapdiagram als het procesfasediagram. Zoals gezegd bestaat het processtapdiagram uit alle coördinatieve en productieve acties die de actoren uitvoeren in een bepaalde interactie. Het is daarbij per processtap (hetgeen een coördinatieve of productieve actie is) mogelijk aan te geven of deze stap optioneel is en of deze stap afhankelijk is van het tot stand

komen van een ander coördinatief feit. Meerdere interacties kunnen ook gezamenlijk in één diagram weergegeven worden. Het procesfasediagram is een meer compacte weergave hiervan waarbij slechts de drie fasen van de transactie gemodelleerd zijn. In figuur C.2 is het voorbeeld van de bibliotheek uitgewerkt in een processtapdiagram.



Figuur C.2: Processtapdiagram bibliotheek, uit [14]

In dit diagram zijn ten eerste wederom de actoren te zien die een rol spelen in de transactie. Een coördinatieve actie die door een actor gedaan wordt is vervolgens aangegeven met een vierkantje, de toestand die dit tot gevolg heeft (een bepaald coördinatief feit) wordt gerepresenteerd door een rondje. In zowel het vierkantje als het rondje staat aangegeven wat de intentie is van de coördinatieve actie. Mogelijke intenties zijn in een transactie *request* (rq), *promise* (pm), *state* (st) en *accept* (ac).

Het verloop van de gehele transactie BT01 is in het diagram te zien. Als gevolg van bijvoorbeeld de (coördinatieve) actie 'request' die S01 doet is het (coördinatieve) feit 'requested' ontstaan. De propositie in deze coördinatieve actie staat niet vermeld, het is per definitie wel zo dat de propositie in de acties die behoren tot dezelfde transactie dezelfde is. Het feit 'requested' is agendum voor actor A01, die hierop vervolgens actie 'promise' doet. Zoals eerder gezegd hebben actoren in een conversatie de mogelijkheid in discussie te gaan en bijvoorbeeld de propositie te weigeren, waardoor de transactie stopt. Dit is hierin niet weergegeven, en wordt normaal gesproken ook weggelaten².

Het feit 'promised' leidt bij actor A01 tot twee acties: ten eerste doet hij een coördinatieve actie (request) gericht aan actor S01. In het coördinatiediagram (figuur C.1) is te zien dat deze actie het verzoeken om betaling van lidmaatschap behelst. Daarnaast doet de actor een productieve actie (vierkantje) die resulteert in een productief feit (grijze ruit). De eigenschap van CAP-net dat elke actie leidt tot precies één feit is

²In [13] is een voorbeeld gegeven waarin deze mogelijkheid wel is uitgewerkt.

hier gebruikt, zodat de actie en het feit samen in één element zijn weergegeven.

Je kunt in de bibliotheek pas lid worden nadat de contributie is betaald. Actor A01 wacht daarom met het doen van zijn productieve actie (het registreren van een nieuw lidmaatschap) totdat de transactie BT02 is afgerond (betaling lidmaatschap). Dit is aangegeven met een stippellijn.

Ter verduidelijking van het voorbeeld is ook met een stippellijn is aangegeven welke acties tezamen een transactie vormen, dit is echter geen onderdeel van het processtapdiagram.

C.3.3 Feitenmodel

De acties die actoren uitvoeren brengen feiten tot stand: de coördinatieve acties brengen coördinatieve feiten tot stand, de productieve acties brengen productieve feiten tot stand. Deze acties zijn toestandsovergangen, zij brengen een verandering tot stand in de coördinatiewereld of de productiewereld. De verzameling van alle tot stand gebrachte feiten in zowel de coördinatie- als de productiewereld vormt de toestand van het systeem. De toestandruimte van het systeem wordt gevormd door alle mogelijke coördinatieve en productieve feiten die tot stand kunnen worden gebracht. Deze toestandruimte wordt beschreven in het feitenmodel. Meestal wordt slechts de toestandruimte beschreven van de productiewereld; de toestandruimte van de coördinatiewereld wordt namelijk al impliciet beschreven in de DEMO-theorie, waarin staat welke coördinatieve acties kunnen plaatsvinden en welk verband deze met elkaar hebben.

De mogelijke toestanden van de productiewereld worden beschreven in een feitenmodel dat veel overeenkomsten heeft met een ORM-model [19]. In een ORM-model worden feittypen gemodelleerd als relaties, verbanden, tussen één of meer rollen. Deze rollen worden gespeeld door objecttypen of waardetypen. Het feittype '*Persoon P heeft voornaam V*' legt een relatie tussen personen en voornamen, en bestaat uit twee rollen: de ene rol wordt gespeeld door het objecttype *Persoon*, de andere rol door het waardetype *Voornaam*.

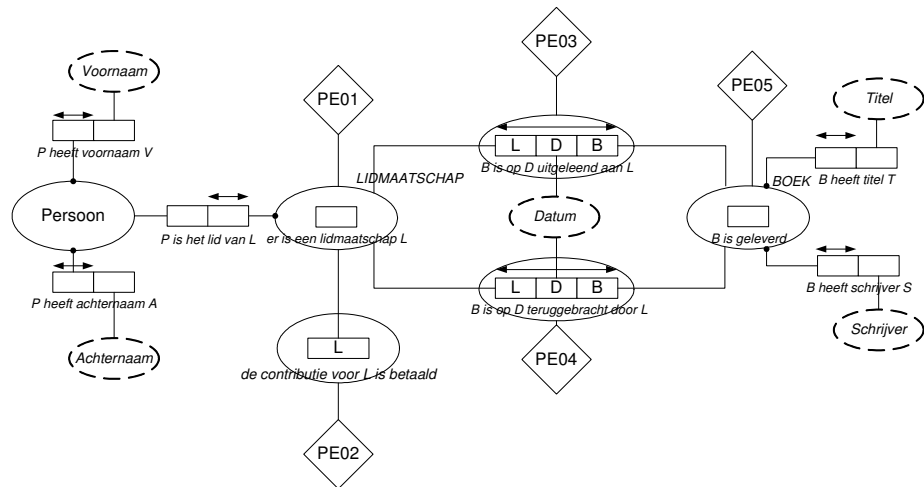
In figuur C.3 is het feitenmodel uit het voorbeeld van de bibliotheek uitgewerkt. Daarin zijn ten eerste een aantal feittypen te zien, die een uitspraak doen over de productiewereld. Deze feittypen bestaan uit één of meer rollen, die gespeeld worden door objecttypen (weergegeven door een ovaal) of waardetypen (gestippelde ovaal). We zien bijvoorbeeld het unaire feittype '*er is een lidmaatschap L*'. Dit feittype doet een uitspraak over een lidmaatschap, namelijk dat het bestaat. We zien ook het feittype '*de contributie voor lidmaatschap L is betaald*'. Dit doet een andere uitspraak over het lidmaatschap, namelijk dat de contributie ervan betaald is. Het feitenmodel bepaalt de structuur van de feiten die in het systeem tot stand worden gebracht. In het feitenmodel zijn feittypen weergegeven; de werkelijke feiten worden gemaakt door deze feittypen te vullen met bepaalde waarden, en zo een populatie van de feittypen aan te maken. Het eerste feittype zouden we bijvoorbeeld kunnen populieren met de feiten '*er is een lidmaatschap #1*', '*er is een lidmaatschap #2*' en '*er is een lidmaatschap #3*'. Het tweede feittype zouden we vervolgens kunnen populieren met de feiten '*de contributie van lidmaatschap #2 is betaald*', hetgeen betekent dat de leden #1 en #3 hun contributie nog moeten betalen.

We kunnen zien dat het eerste feittype tot stand wordt gebracht door transactie T01: dit weergegeven door de ruit met daarin het eventtype PE01. Dit betekent dat het resultaat van transactie T01 bestaat uit het feittype '*er is een lidmaatschap L*', of anders gezegd, dat transactie T01 een lidmaatschap creëert. Een lidmaatschap wordt aangemaakt in de executiefase van transactie T01 (dan vindt de productieve actie plaats),

maar het bestaat pas werkelijk als de transactie in de toestand *geaccepteerd* is gekomen.

Het feitenmodel definieert de structuur van de feiten, de populatie daarvan zijn de werkelijke feiten, die een uitspraak doen over de productiewereld. Het feitenmodel bepaalt de mogelijke feiten, en definieert dus de toestandsruimte (welke toestanden er mogelijk zijn). De populatie daarvan geeft de werkelijke toestand (op een bepaald moment) van het systeem weer.

Daarnaast is het mogelijk om beperkingen op te leggen aan de populatie van een feittype. In het feittype '*Persoon P is het lid van Lidmaatschap L*' zien we dat aan de rol van lidmaatschap in dit feittype een zwarte stip is getekend, hetgeen een totaliteitsconstrait weergeeft. Dit betekent dat alle lidmaatschappen die er bestaan een rol moeten spelen in de relatie '*Persoon P is het lid van Lidmaatschap L*' ; anders gezegd: van elk lidmaatschap moet gezegd worden welk persoon daarbij hoort. Ook zien we dat boven de rol lidmaatschap in dit feittype een streep staat. Dit geeft een uniciteitsconstrait weer, hetgeen betekent dat elke instantie van lidmaatschap hoogstens één keer in de relatie mag voorkomen: het is dus niet toegestaan om te zeggen '*Persoon #1 is het lid van Lidmaatschap #2*' en ook te zeggen '*Persoon #3 is het lid van Lidmaatschap #2*'. Het lidmaatschap #2 mag maar één keer in de populatie van dit feittype voorkomen: een lidmaatschap hoort dus toe aan maximaal één persoon. De combinatie van de totaliteitsregel en de uniciteitsregel op dit feittype betekent vervolgens dat een lidmaatschap toevoert aan precies één persoon, niet meer (uniciteit) en niet minder (totaliteit).



Figuur C.3: Feitendiagram bibliotheek