Een kritische analyse van agenttechnologie als ondersteuning voor logistieke netwerken

Mario AERTS

promotor :
Prof. dr. Gerrit JANSSENS

co-promotor :
Prof. dr. Koen VANHOOF

Eindverhandeling voorgedragen tot het bekomen van de graad
Handelsingenieur in de beleidsinformatica

universiteit hasselt

Woord vooraf

Het opleveren van deze thesis is één van de hoogtepunten van mijn vijf jaar durende opleiding tot Handelsingenieur in de Beleidsinformatica. Gedurende een jaar verdiepte ik mij in de interessante materie van agenttechnologie. Gezien de relatieve jeugdigheid van het concept agenttechnologie vormde het doorgronden en bestuderen van deze materie een heuse uitdaging. Het afwerken van deze analyse van agenttechnologie in functie van logistieke netwerken is dan ook met de gepaste voldoening gebeurd.

Met deze voldoening heb ik ook de rest van mijn opleiding voltooid. Ik bedank dan ook mijn ouders voor het advies en de grote vrijheid die ze mij gaven in de keuzes die ik diende te maken in mijn leven. Zonder hun steun zou ik dus nooit geraakt zijn waar ik nu sta in mijn jonge leven.

Verder is toch ook een woord van dank gepast voor mijn promotor prof. dr. G. Janssens en mijn co-promotor prof. dr. K. Vanhoof die instonden voor de begeleiding van deze eindverhandeling. In deze context wil ik ook prof. dr. Holvoet, dr. ir. Roos en de heer Linten bedanken voor hun leerrijke en verdiepende interviews. In het bijzonder wil ik ook meneer Tijs Vermeulen van de vakgroep informatietechnologie van de Katholieke Hogeschool Sint-Lieven te Gent onder leiding van prof. dr. Greet Vanden Berghe bedanken voor de deskundige introductie in het MamMoeT-platform en de aangename manier waarop hij mij meermaals hielp met het beantwoorden van mijn vragen. Een bijzonder dankwoord gaat ook uit naar een speciaal iemand voor mij, namelijk Evy Cuypers die niet alleen een belangrijke uitlaatklep was in moeilijke tijden maar ook op toegewijde manier al mijn teksten nalas.

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

Samenvatting

De logistieke sector is de laatste jaren blootgesteld aan verschillende invloeden die een groot effect hebben gehad op de manier van zakendoen in deze sector. Duidelijk herkenbare invloeden zijn de globalisatie van de wereldeconomie, de veranderingen in traditionele organisatiemodellen en de opgang van het multimodale vervoer. Bovendien hebben verladers steeds hogere eisen op het gebied van flexibiliteit, snelheid, efficiëntie, betrouwbaarheid, transparantie en connectiviteit.

Als reactie op deze veranderingen gingen logistieke spelers zich meer en meer concentreren op hun kernactiviteit om op deze manier competitief voordeel te bekomen op de markt. Toch vereist de verlader een geïntegreerde keten zodat vele logistieke dienstverleners onderling nauw dienen samen te werken om bepaalde processen te voltooien. De verzameling van deze dienstverleners die er onderling complexe relaties op nahouden wordt ook wel een logistiek netwerk genoemd. De noodzaak aan intensieve communicatie binnen dergelijk netwerk is dus groter dan ooit.

De communicatie tussen logistieke spelers stelt hoge eisen aan de ondersteuning die Informatie Technologie (IT) dient te bieden voor de sector. Een overzicht van actuele toepassingen binnen de logistieke sector leert ons dat de huidige bestaande toepassingen niet geschikt zijn om gedistribueerde processen te ondersteunen. Samen met geavanceerde planningsprogramma's en online platformen kunnen ERP-systemen en EDI immers niet de nodige flexibiliteit aan de dag leggen om de complexiteit van de relaties binnen een logistiek netwerk ten volle te ondersteunen.

Agenttechnologie is een nieuwe manier om een softwareapplicatie te structureren rond autonome en communicatieve elementen, ook wel agenten genoemd. In de literatuur geraakt men het er moeilijk over eens wat een agent nu werkelijk is. Wel vinden we beschrijvingen terug van de eigenschappen die agenten bezitten. Een vereiste is dat een agent een zekere mate van autonomie bezit. Deze autonomie betekent dat een agent, in tegenstelling tot objecten die we kennen van het objectgeoriënteerde paradigma, de volledige controle over zijn toestand en acties heeft. Om interactie met andere agenten mogelijk te maken, dient een agent ook over een sociale dimensie te beschikken. Deze dimensie bepaalt de graad van complexiteit die kenmerkend is voor de

berichtuitwisseling van agenten. In het beste geval kunnen twee agenten coöperatief zijn zodat ze onderling kunnen negotiëren en voor een bepaald probleem de beste oplossing selecteren. Een laatste kritische eigenschap van agenten is hun doelgericht gedrag. We kunnen stellen dat een agent in het uitvoeren van dit gedrag op zijn minst een zekere graad van reactiviteit en pro-activiteit dient te bezitten. Verder kunnen we stellen dat een agent een zekere graad van intelligentie bezit. Deze eigenschap wordt samen met mobiliteit als secundaire eigenschap beschouwd en is dus geen kritische eigenschap van een agent.

Hoewel agenten ook als individuele applicaties gebruikt kunnen worden in de vorm van intelligente assistenten van gebruikers, wordt er hoofdzakelijk ingegaan op de rol van agenten binnen multi-agentsystemen. Zogenaamde expertagenten hebben toegang tot gespecialiseerde data en modelleren een bepaald domein in de werkelijkheid waarbinnen zij een bepaalde verantwoordelijkheid bekleden. Naast deze expertagenten die hoofdzakelijk gespecialiseerde diensten aanbieden aan medeagenten zijn er transactieagenten die hoofdzakelijk instaan voor het afhandelen en organiseren van interacties met andere agenten. Een derde soort agenten zijn agenten die op systeemniveau enkele taken voorzien en op deze manier de werking van het multi-agentsysteem vergemakkelijken. Tot slot zijn er de informatie- of webagenten die heterogene data kunnen beheren en presenteren aan de gebruiker of andere agenten.

Aangezien een multi-agentsysteem samengesteld is uit verschillende agenten is het een componentgebaseerde softwareapplicatie met een peer-to-peer architectuur. Via het agentparadigma wordt een probleemdomein door modularisatie en abstractie opgedeeld in verantwoordelijkheden die eigen zijn aan agenten. De gedistribueerde procesafhandeling binnen een multi-agentsysteem wordt ondersteund door asynchrone berichtuitwisseling tussen agenten. Deze berichtuitwisseling die gestuurd wordt door systeemcomponenten gebeurt via een Agent Communication Language (ACL). Naast regels op syntactisch niveau worden in ACLs semantische regels gedefinieerd. Elk ACL-bericht zal dan gekenmerkt worden door een ‘communicative act’ die de activiteiten die agenten uitvoeren en de uitgewisselde berichten op elkaar afstemmen. Het is natuurlijk evident dat dergelijke concepten gepaard moeten gaan met bepaalde afspraken zodat er een standaardisatie bekomen wordt. Op deze wijze kunnen agentsystemen immers interoperabel worden.

De nood aan standaardisatie binnen de agentwereld werd onder andere ingevuld door FIPA. FIPA heeft naast hun populaire ACL een set aan standaarden ontwikkeld die hun abstracte specificatie van een multi-agentsysteem concretiseren. Dit wordt bewerkstelligd door een specificatie van zowel de individuele componenten als de interacties en verbanden tussen deze componenten. Een analyse van deze FIPA standaarden leert ons echter dat andere standaarden om een integratie aan softwarecomponenten te bewerkstelligen meer bijval genieten van de industrie. We denken hier dan specifiek aan Web Services. Verder is er nog veel werk aan de winkel vooral rond het integreren van een gestandaardiseerd mechanisme voor gebruik van ontologieën in ACL-berichten. Op dit gebied kan het Semantic Web waarschijnlijk een grote rol gaan spelen.

Het voldoen aan standaarden samen met de integratie van ondersteunende technologieën als P2P computing, internet- en transportprotocollen en berichtformaten zoals XML, is een grote last op de schouders van de programmeur. Middleware-platformen waarin dergelijke functionaliteit reeds vervat zit en die makkelijk gebruikt kan worden, zijn dus een belangrijk hulpmiddel voor de programmeur. In deze context wordt het JADE platform bestudeerd dat voldoet aan de FIPA-standaarden en volledig ontwikkeld is in Java. Over de toepasbaarheid van JADE in logistieke netwerken dienen we echter ook kritisch te zijn. Zo wordt het ernstig in twijfel getrokken of JADE kan voldoen aan de complexe eisen van de industrie daar de eisen voor de meeste logistieke toepassingen te specifiek zullen zijn. Toch is JADE een belangrijk platform voor 'proof-of-concept' implementaties waar geëxperimenteerd kan worden met agentarchitecturen en designspecificaties.

De selectie- en analysefase voor een softwareproject die uitgewerkt worden in een concrete case bij het Europees Distributiecentrum van SKF te Tongeren illustreren de mogelijke ondersteuning die agenttoepassingen kunnen bieden binnen logistieke netwerken. In die zin overlapt de case de bevindingen uit de literatuur. Hoewel de kenmerken en kwaliteiten van agentsystemen dus uiterst geschikt zijn als ondersteuning voor logistieke netwerken, is het op dit moment dus nog gissen naar de manier waarop deze kenmerken en kwaliteiten geïmplementeerd zullen worden in toepassingen voor de industrie.

HOOFDSTUK 1: Inleiding

1.1 Situering

De laatste decennia zijn de relaties tussen de organisaties in transportketens ingrijpend veranderd. Meer en meer maken de eenduidige en hiërarchische relaties tussen logistieke spelers, die kenmerkend waren voor traditionele transportketens, plaats voor ingewikkelde netwerken. (Bitran et al., 2006) In de literatuur gebruikt men dan ook steeds meer de term logistiek netwerk (Verduijn, g.d.). Een logistiek netwerk is als een verzameling van logistieke spelers die samen in staat zijn om meerdere verschillende logistieke ketens te vormen.

Het is duidelijk dat de concepten waardeketen of supply chain gelinkt zijn aan het nieuwe concept van een logistiek netwerk. Een waardeketen is een groep van organisaties die een samenhang vertonen door stroomopwaartse en stroomafwaartse relaties binnen verschillende processen en activiteiten uitvoeren die waarde toevoegen aan producten of diensten (Christopher, 2000). Powell (1992) definieert in Verduijn et al. (2003) een netwerk als volgt: "A network is a mode of resource allocation where transactions occur neither through discrete exchanges nor by administrative fiat but through networks of individuals engaged in reciprocal, preferential, mutually supportive actions". De relaties tussen verschillende actoren in deze netwerken zijn dus niet eenduidig bepaald en erg variabel naargelang de context waarin de actoren zich bevinden. De rol die bedrijven invullen in een specifieke waardeketen is afhankelijk van de opdrachten die andere bedrijven aan hen uitbesteden. Het is vaak zelfs zo dat bedrijven bij de uitvoering van die opdracht bronnen gebruiken die ter beschikking gesteld zijn door het andere bedrijf. Als illustrerend voorbeeld halen we hier bijvoorbeeld de wegvervoerder aan die de containers van de rederij vervoert naar het binnenland.

1.1.1 Invloeden op de logistieke sector

In het algemeen wijdt men het complexer worden van het logistieke landschap aan een drietal oorzaken: de globalisatie van de wereldeconomie, de veranderingen in traditionele organisatiemodellen en de opgang van het multimodale vervoer. De evolutie naar een globale economie brengt met zich mee dat ondernemingen op globaal niveau met elkaar samenwerken en/of elkaar beconcurreren (Matthyssens et al., 1998; PwC consulting, 2002). Bedrijven willen competitief blijven op de globaliserende markt. Logistieke spelers die dus competitief willen zijn en blijven, zullen noodzakelijkerwijs moeten concurreren met organisaties wereldwijd. Dit betekent voor vele organisaties dat zij hun logistieke dienstverlening noodgedwongen binnen een groter geografisch gebied moeten gaan aanbieden.

Bedrijfsmodellen zijn onderhevig aan verandering. Het is immers zo dat de nadruk die lag op kapitaalintensieve activiteiten zoals het oprichten van productiefaciliteiten en nieuwe vestigingen stilaan is verminderd. De nadruk verschuift naar minder kapitaalintensieve activiteiten zoals productiemiddelen en -processen en naar activiteiten die de kracht van hun merk doen toenemen. In deze nieuwe businessmodellen zijn 'human capital' en 'brand capital' kritische succesfactoren. Het is evident dat deze verschuiving naar nieuwe businessmodellen repercussies heeft voor alle processen in de onderneming. (PwC consulting, 2002) Dit betekent dus dat ondernemingen in mindere mate hun logistieke processen zelf zullen beheren. In plaats daarvan gaan ze deze processen 'outsourcen' naar logistieke dienstverleners.

Naast de globalisatie is ook het stijgende belang van het multimodale transportmodel een factor die de logistieke omgeving in zekere mate complexer maakt. In een multimodale keten zullen meerdere transportmodi het transport van de goederen verzorgen. Waar goederen vroeger bijvoorbeeld enkel via vrachtvervoer getransporteerd werden in het binnenland zal dit transport meer en meer gecombineerd gaan verlopen via trein, binnenschip of vrachtwagen. Op deze manier wordt het logistieke netwerk gevoelig uitgebreid omdat er binnen de keten zowel vertegenwoordigers van het wegvervoer als van het spoor of de binnenvaart een rol gaan spelen. In een volgend hoofdstuk zal de stijgende populariteit, de opbouw en de relaties van dergelijk (multimodaal) logistiek netwerk geschetst worden.

Bovenstaande evoluties gaan gepaard met steeds hogere eisen van de verladers op gebied van flexibiliteit, snelheid, efficiëntie, betrouwbaarheid, transparantie en connectiviteit (Rome, 2004; PwC consulting, 2002). Al deze kwalitatieve aspecten hebben een dwingende invloed tot verandering op het logistieke landschap. Door specialisatie en outsourcing als antwoord op de geschetste evoluties worden traditionele logistieke businessmodellen vervangen door verticaal geïntegreerde organisaties die bestaan uit verschillende partners in het logistiek netwerk. Outsourcing is het fenomeen waarbij de activiteiten en verantwoordelijkheden die samenhangen met bepaalde processen uitbesteed worden aan externe firma's. Via deze praktijk kan een bedrijf zich concentreren op zijn kernactiviteiten en een competitief voordeel creëren dat gepaard gaat met verminderde kosten (Chase et al., 2006). Rome (2004) zegt hierover: "elke partner focust op zijn competentiedomein of op zijn basisregio en samen bieden de coöpererende partijen geïntegreerde diensten aan". Verder lezen we bij hem dat deze outsourcing-trend zeer prominent aanwezig is in het domein van de logistiek.

Voorgaand betoog toont ons duidelijk in welk complex landschap een logistieke speler zich bevindt. Het is duidelijk dat een nauwe samenwerking met andere logistieke spelers zich opdringt. Bovendien is de mogelijkheid om competitief voordeel te behalen ten opzichte van concurrenten niet alleen gebaseerd op de prestaties van de individuele firma maar ook de flexibiliteit, efficiëntie en effectiviteit van de supply chain (Christopher, 1992: geciteerd door Verduijn et al., 2003). Macharis (2004) zegt hierover dat vele organisaties hierin slagen dankzij strategische allianties, overnames of samenwerkingsakkoorden.

Deze consolidatietrend wordt gedreven door een drietal factoren. Door deze samenwerking realiseren bedrijven schaalvoordelen in het gebruik van hun materiële activa zoals onder andere vrachtwagens, magazijnen. Vervolgens kunnen samenwerkende partners profiteren van elkaars relaties en ervaringen op bepaalde domeinen. In het rapport van PwC consulting refereert men hiervoor naar het concept van 'economies of skill/scope'. Een derde belangrijke factor is het verhogen van klantenservice. Verladers willen immers logistieke dienstverleners die het volledige logistieke proces voor hun rekening nemen, inclusief de coördinatie van de logistieke keten. (PwC consulting, 2002) Rome (2004) vermeldt dat bedrijven hiermee de strategie van 'one-stop shopping' nastreven en daardoor in staat zijn quasi alle componenten van de logistieke keten aan te bieden.

1.1.2 De logistieke supply chain

De nauwe samenwerking tussen onderlinge schakels in een supply chain kan enkel tot stand komen door voldoende aandacht te schenken aan het supply chain management (SCM). Laudon en Laudon (2005) definiëren SCM als volgt: "SCM refers to the close linkage and coordination of activities involved in buying, making, and moving a product. It integrates business processes to speed information, product, and fund flows up and down a supply chain to reduce time, redundant effort, and inventory cost". Binnen SCM speelt het logistieke gedeelte een belangrijke rol. De logistieke supply chain vormt een belangrijk onderdeel van de totale supply chain. De logistieke waardeketen moet de planning en de controle afhandelen van alle factoren die een invloed uitoefenen op het transport. Dit transport staat in voor het verplaatsen van het juiste goed, naar de juiste plaats, in zo weinig mogelijk tijd, voorzien van de juiste toegevoegde diensten en aan de minste kost (Laudon en Laudon, 2005). Vergeten we niet dat de logistieke spelers binnen één logistieke waardeketen ook deel uit kunnen maken van andere waardeketens. In deze andere waardeketens hebben zij dan mogelijkerwijs relaties met andere spelers uit hun logistiek netwerk.

Het hoeft geen betoog dat het afstemmen van de processen binnen een logistieke waardeketen moet gebeuren door een zorgvuldige coördinatie (Gunasekaran en Ngai, 2003). De coördinatie van de supply chain hangt sterk samen met de mogelijkheid en kwaliteit van het uitwisselen van informatie. Rome (2004) stelt zelfs dat de informatiestroom tussen de diverse spelers in een distributieketting meestal belangrijker en bedrijfskritischer is dan de goederenstroom.

1.1.3 ICT en de logistieke supply chain

Het samenwerken binnen de supply chain over de bedrijfsgrenzen heen wordt mogelijk gemaakt door Informatie en Communicatie Technologie (ICT of IT). Vele onderzoekers hebben reeds de nauwe link tussen informatiesystemen en het managen van het logistieke proces aangetoond. ICT maakt het mogelijk om informatie snel, juist en goedkoop te verspreiden en zorgt dus voor een betere coördinatie binnen de keten en een efficiënter SCM (Gunasekaran en Ngai, 2003). We kunnen zelfs verder gaan en stellen dat het zonder ICT onmogelijk is om business te doen in een dergelijke omgeving

waar communicatie en coördinatie tussen externe partijen broodnodig is. De verwoording van Gunasekaran en Ngai (2003) ondersteunt dit: "IT is like a nerve system for SCM". Deze verbetering van de efficiëntie wordt bewerkstelligd door de ondersteuning die IT toepassingen kunnen bieden voor zowel de planning als de uitvoering van de supply chain. Verder zorgt ICT voor een verbetering van de onderlinge connecties tussen de partners in de keten. (Evangelista, g.d.)

Zoals we in een volgend hoofdstuk zullen aantonen, zijn traditionele technologieën binnen de ICT-wereld niet toereikend genoeg om binnen de geschetste complexe organisatiestructuur te kunnen functioneren. Geïsoleerde toepassingen die gericht zijn op het intern afhandelen van processen binnen één organisatie kunnen niet voldoende omspringen met de dynamische en gedistribueerde aard van processen en beslissingsprocedures binnen de logistieke supply chain (Verduijn et al., 2003). Zoals reeds aangehaald is de huidige situatie van een logistieke actor nog complexer omdat hij zich in een logistiek netwerk bevindt. Naargelang de situatie zal hij moeten samenwerken met verschillende spelers uit het logistieke netwerk waarmee hij de processen binnen een bepaalde supply chain verzorgd. Met technologieën als Electronic Data Interchange (EDI) en systemen voor Enterprise Resource Planning (ERP) en SCM slaagt men er niet in om de hoge eisen van de complexe en dynamische omgeving werkelijk te integreren in een gedistribueerde ICT-toepassing. Het is daarom dat binnen de transportwereld een volledige fragmentering van data en incompatibiliteit van informatiesystemen terug te vinden is (Macharis, 2004).

1.2 Agenttechnologie

De ICT-sector is de afgelopen decennia gekenmerkt door haar vliegensvlugge evolutie. Nieuwe standaarden en verbeterde technologieën zorgen voor een betere basisinfrastructuur waar toepassingen gebruik van kunnen maken. Deze technologieën zullen verder in de thesis nog uitgebreid besproken worden. We kunnen nu toch al aanhalen dat ze de mogelijkheid bieden om de aandacht te verschuiven van individuele 'stand-alone' applicaties naar gedistribueerde netwerken van applicaties die samenwerken om een proces uit te voeren (Langley, 2003). In het licht van deze trend is het afgelopen decennium veel energie en budget gestoken in het onderzoek naar agenttechnologie. Agenttechnologie is een nieuwe manier om een softwareapplicatie te

structureren rond autonome en communicatieve elementen, ook wel agenten genoemd (Jennings et al., 1998). De kenmerken van individuele agenten, die nog meer gedetailleerd aan bod zullen komen, zorgen ervoor dat agenttechnologie een belangrijk en nieuw paradigma is dat gebruikt kan zijn om applicaties te ontwikkelen.

1.2.1 Vergelijking met objectoriëntatie

Omdat agenttechnologie een belangrijk nieuw paradigma is om programma's te ontwikkelen wordt het ook wel eens vergeleken met de objectgeoriënteerde structuur van programma's. Via objectgeoriënteerd programmeren is men in de jaren '90 tegemoet gekomen aan steeds hogere eisen van gebruikers. Deze eisen leidden onverbiddelijk tot grote en onoverzichtelijke software projecten die moeilijk te beheren waren. Objectgeoriënteerd ontwikkelen loste zowel problemen van gebruikers als van ontwikkelaars op. Deze objectgeoriënteerde aanpak doemde de sequentiële en functionele aard van programma's naar de verleden tijd en zorgde voor een revolutie in softwareplanning en -ontwikkeling.

Objecten zijn software-entiteiten die bepaalde waarden of datastructuren opslaan in variabelen en methoden voorzien waardoor deze waarden aangepast of opgevraagd kunnen worden door andere objecten. Men zegt ook wel dat data door objecten wordt ingekapseld. Op deze manier kan de werkelijkheid gemodelleerd worden doordat instanties van bepaalde klassenobjecten onderling elkaars methoden gaan aanroepen. Bij deze modulaire opbouw neemt de herbruikbaarheid van code toe. Toch blijven de variabelen enkel toegankelijk binnen de instanties van de klassen waar ze gedeclareerd zijn. Op deze manier is een programma dat via goede objectoriëntatie ontwikkeld is veel robuuster en overzichtelijker omdat elke klasse zo ontwikkeld is dat ze instaat voor een specifieke functie.

Agenttechnologie is in dit opzicht vernieuwend daar het verder gaat dan het objectgeoriënteerde paradigma. Binnen objectoriëntatie hebben objecten geen controle over de aanroeping van de methoden die zij definiëren. Zij zijn dus als het ware een passieve uitvoerder van de aanroepen die andere objecten doen op hun methodes. Agenten daarentegen zijn autonoom wat wil zeggen dat ze de volledige controle over de aanroeping van hun eigen methoden hebben. Agenten vertonen een individueel

doelgericht gedrag en gaan op basis van de omgevingsstimuli bepaalde beslissingen nemen om hun eigen methodes uit te voeren. (Odell, 2004; Lynch en Rajendran, 2005) In die zin zal een vragende agent een bepaalde methode van een uitvoerende agent niet zonder meer kunnen aanroepen maar eerder een aanvraag doen aan die agent om de methode uit te voeren (Jennings et al., 1998). De beslissing of de methode uitgevoerd gaat worden ligt uiteindelijk altijd bij de agent die controle heeft over de methode. In feite zijn agenten objecten die uitgebreid zijn met enkele menselijke kenmerken of zoals Bradshaw (1997: geciteerd door Odell, 2004) over agenten zegt: "objects with an attitude". Het bezitten van autonomie wordt in de literatuur en in de afgenomen interviews als dé belangrijkste eigenschap van een agent beschouwd.

Tot slot kunnen we stellen dat de interne toestand van een agent bepaald wordt door een complexer geheel van elementen dan de toestand van een object. De toestand van een object werd bepaald door datatypes die belangrijk waren voor het verantwoordelijkheidsdomein van een object. De interne toestand van een agent daarentegen staat in functie van een doel en omvat meer diepgaande concepten zoals plannen, concepten en interpretaties van de werkelijkheid. Dit laatste wordt in de literatuur ook wel de 'believes' van een agent genoemd of datgene wat hij gelooft. (Bauer, 2002)

1.2.2 Beschrijving van de modulaire eenheid van agenttechnologie: de agent

Bij het doornemen van de literatuur betreffende softwareagent gerelateerde onderwerpen valt dadelijk op dat een universeel geaccepteerde definitie van een agent niet bestaat. Toch komen in de omschrijvingen een aantal steeds terugkerende elementen terug. Zo kunnen we stellen dat een agent een elektronische entiteit is met menselijke karakteristieken die gesitueerd is binnen een complexe, dynamische omgeving en in staat is om op een flexibele en autonome manier acties te ondernemen om zijn vooraf bepaalde doelen te bereiken. Deze doelen proberen agenten te bereiken door het vertegenwoordigen van de belangen van de gebruiker of een andere agent die een dienst vraagt. Agenten reageren op impulsen uit hun omgeving die ze als input kunnen gebruiken om de omgeving op één of andere manier te veranderen. Vaak veronderstellen auteurs dat er bij dit proces een zekere redenering gebeurt op basis van bepaalde kennis die voorhanden is. Dit impliceert dat de agent een zekere intelligentie bezit. Vandaar dat

men in de literatuur ook vaak de term 'intelligent agent' hanteert in plaats van gewoon 'agent'. (Jennings et al., 1998; Franklin en Graesser, 1996; Keele en Wray, 2005; AgentLink III, 2005)

We zullen even verder ingaan op de omschrijving van een agent. Flexibel zijn in acties slaat op responsiviteit, pro-activiteit en het sociale karakter van een agent. Volgens deze stelling bezit een agent dus niet enkel responsiviteit of reactiviteit, dit is de mogelijkheid van een agent om zijn omgeving te percipiëren en hierop te reageren, maar ook pro-activiteit. Met dit laatste bedoelen we dat agenten niet enkel als antwoord op een impuls in hun omgeving een actie uitvoeren maar ook opportuniteiten kunnen aangrijpen door assertief gedrag. Agenten kunnen dus zelfstandig acties ondernemen waarvan de uitwerking een bijdrage levert aan hun doelgericht gedrag. Tot slot heeft de flexibiliteit een sociale dimensie. Binnen multi-agentsystemen is deze sociale dimensie onafwendbaar en erg belangrijk. Het sociaal zijn van agenten slaat op de mogelijkheid om interactie met andere elektronische agenten te kunnen hebben teneinde hun eigen doel te bereiken of anderen te helpen met hun activiteiten. (Jennings et. al. 1998; Franklin en Graesser, 1996; Keele en Wray, 2005; AgentLink III, 2005)

Als laatste en belangrijkste aspect bezit een agent autonomie. Hiermee wordt bedoeld dat een agent de volledige controle over zijn acties en interne toestand moet hebben en zijn functies kan vervullen zonder de interventie van mensen. Autonomie en intelligentie worden bij enkele auteurs ook gelinkt aan 'artificial intelligence' (AI). Zij veronderstellen dat agenten zelfstandig in staat zijn om te leren uit omgevingselementen en acties die zij in het verleden uitgevoerd hebben. (Jennings et. al. 1998; Franklin en Graesser, 1996; Keele en Wray, 2005; AgentLink III, 2005)

Gezien de diverse opvattingen betreffende agenten in wetenschappelijke kringen wordt bovenstaande beschrijving van agenten die autonoom, sociaal, reactief en proactief zijn ook wel bestempeld als 'a weak notion of agency' (een zwakke opvatting over agenten) omdat het een eerder oppervlakkige en basisopvatting is van de kenmerken van een agent (Jennings en Wooldridge, 1994). Anderen houden er een 'strong notion of agency' op na waarmee zij naast de kenmerken die we in de zwakke vorm terugvinden ook nog intelligentie en mobiliteit van agenten in rekenschap nemen.

Agenten zijn dus een vorm van complexe objecten die gebaseerd zijn op het objectgeoriënteerd paradigma. Deze verhoogde complexiteit dient benaderd te worden vanuit een zo breed mogelijk perspectief. Daarom zijn ook een aantal andere wetenschappen betrokken in het onderzoek naar en ontwikkelen van agentgebaseerde systemen. "Agent technology stems from a convergence of many disciplines from artificial intelligence to software engineering to economic and social sciences" (Gelati, 2004). Deze verzameling van wetenschappen hebben ertoe bijgedragen dat agenttechnologie technieken en algoritmes omvat waardoor agenten kunnen omgaan met zowel menselijke als elektronische agenten in dynamische en open omgevingen. Verder kan men via deze technologie in staat zijn om bepaalde mechanismen te automatiseren en reeds bestaande processen te verbeteren waarvan men vroeger dacht dat dit onmogelijk was. Verder zouden applicaties die volgens agenttechnologie ontwikkeld zijn, kunnen anticiperen en handelen naargelang de behoeften van de gebruiker. Agenten zijn dus ook in staat om de belangen van de gebruiker, al dan niet menselijk, te vertegenwoordigen (Luck et al., 2004).

Bovenstaande beschreven kenmerken zorgen ervoor dat agenten gebruikt kunnen worden door ontwikkelaars om grote, complexe systemen op te bouwen met agenten als bouwstenen. Het grote en complexe systeem kan opgebroken worden in kleine subsystemen die samenhangen met de functies van één agent. Net zoals in de realiteit hebben de agenten onderling complexe relaties die ze kunnen onderhouden door hun sociale mogelijkheden en die gedefinieerd zijn door de interne status en doelgeoriënteerde karakter van de agenten. Dergelijk systeem noemt men ook wel een multi-agentsysteem. (Jennings et. al., 1998; Franklin en Graesser, 1996; Keele en Wray, 2005; AgentLink III, 2004)

1.3 Onderzoeksopzet

1.3.1 Centrale onderzoeksvraag

De beschrijving die we gegeven hebben van agenttechnologie laat vermoeden dat deze technologie uiterst geschikt is als ondersteuning van een logistiek netwerk. De centrale onderzoeksvraag uit deze thesis zal dan ook zijn:

Welke mogelijkheden en toepassingen kan agenttechnologie bieden aan logistieke bedrijven om binnen hun netwerk de logistieke waardeketens te optimaliseren?

1.3.2 Deelvragen

De vermelde centrale onderzoeksvraag bevat twee kritische elementen die nader toegelicht dienen te worden, namelijk logistieke netwerken en agenttechnologie. Zoals we reeds vermeld hebben zijn logistieke netwerken ontstaan door verschillende invloeden waaraan de logistieke sector onderhevig was. Om een analyse te kunnen maken van de rol die agenttechnologie kan spelen voor de logistieke sector dienen we eerst een gedegen beeld te krijgen van een logistiek netwerk en de IT-toepassingen die het meest frequent gebruikt worden. De onderzoeksvraag die hier logischerwijs uit voortvloeit luidt als volgt:

Welke kenmerken hebben logistieke netwerken en hoe worden ze binnen de huidige bedrijfskundige context ondersteund door IT-toepassingen?

Agenttechnologie is een relatief nieuw begrip in de IT-wereld dat nog volop bestudeerd wordt door onderzoeksgroepen over heel de wereld. Vermoedelijk zijn weinig personen vertrouwd met dit begrip. De kenmerken van agenten en multi-agentsystemen zijn merkbaar verschillend van de kenmerken van andere IT-infrastructuren. Een volgende onderzoeksvraag die we kunnen stellen is dan ook:

Welke kenmerken bezitten agenten en multi-agentsystemen en hoe onderscheiden zij zich van de actuele toepassingen binnen de logistieke sector?

Door de kenmerken van agentsystemen te vergelijken met actuele toepassingen heeft men nog altijd geen zicht op de toepasbaarheid van agenttechnologie. Zoals we zullen zien zijn er belangrijke realisaties gebeurd op gebied van standaardisatie van agenttechnologie. Het is van belang dat deze standaarden onder de loep genomen worden. De onderzoeksvraag die we hieraan koppelen is:

Op welke manier zijn aspecten van agenttechnologie gestandaardiseerd en in welke mate beïnvloeden zij de toepasbaarheid van agenttechnologie?

1.3.3 Methodologie

In deze thesis is ervoor gekozen om de kenmerken van logistieke netwerken en agenttechnologie te onderzoeken op basis van een literatuurstudie in combinatie met interviews met bevoorrechte getuigen.

Hoewel op conceptueel niveau het belang van de studie over logistieke netwerken van minder belang is dan de studie van agenttechnologie, heb ik er toch voor gekozen om deze resultaten uit de literatuurstudie te verifiëren aan de hand van getuigenissen uit het bedrijfsleven. Deze verificatie werd bewerkstelligd door een interview met de heer M. Linten, regionaal verantwoordelijke voor Domestic Transport in het Verenigd Koninkrijk en de Benelux van het Europees Distributie Centrum van SKF te Tongeren.

Naast een literatuurstudie over de kenmerken van agenttechnologie voer ik een studie uit van concrete realisaties die een behoorlijke populariteit en acceptatie kennen binnen de agentwereld. In dit geval werden de FIPA-standaarden en het JADE-platform dat een implementatie is van de FIPA-standaarden onder de loep genomen. Na de studie en rapportering van de realisaties werden deze kritisch geanalyseerd, enerzijds aan de hand van interviews met experten op gebied van agenttechnologie en anderzijds met recente reacties in de literatuur, nieuwsbrieven en weblogs van bekende onderzoekers in de agentwereld. Naast een interview met prof. dr. T. Holvoet, verantwoordelijk voor onderzoek naar agenttechnologie binnen de Distributed Systems and Computer Networks (DistriNet) onderzoeksgroep aan de K. U. Leuven, had ik ook een gesprek met dr. ir. N. Roos van het Institute for Knowledge and Agent Technology (IKAT) aan de Universiteit Maastricht.

Verder kon ik rekenen op de bereidwillige medewerking van de vakgroep informatietechnologie van de Katholieke Hogeschool Sint-Lieven te Gent onder leiding van prof. dr. G. Vanden Berghe en vertegenwoordigd door de heer T. Vermeulen, medeontwikkelaar van het agentplatform MamMoeT. Via deze laatste kreeg ik een demonstratie van en had ik toegang tot de code van MamMoeT, een multi-agentsysteem dat in staat is om autonoom vrachten te negotiëren en te boeken voor de binnenvaart. De uitwisseling van ervaring met het ontwikkelen van agentsystemen en het MamMoeT-platform zelf waren voor mij een grote bron van informatie om een beeld te vormen van agentsystemen en het ontwikkelen en implementeren van agentsystemen. Alhoewel ik slechts drie interviews afgelegd heb betreffende agenttechnologie denk ik toch dat de reacties en ervaringen die ik hiermee heb losgeweekt erg leerrijk en representatief zijn voor de agentwereld. Bij het afnemen van interviews dient er immers rekening gehouden te worden met geografische beperkingen zodat het aantal agentexperten dat in aanmerking kwam om te interviewen drastisch daalde. Toch is deze geografische beperking deels opgeheven door het feit dat ik op zoek ging naar actuele discussies en informatie via een nieuwsbrief, namelijk agents@cs.umbc.edu, en weblogs van gekende onderzoekers.

Tot slot koos ik ervoor om de toepasbaarheid van multi-agentsystemen binnen logistieke netwerken te illustreren aan de hand van een concrete case. Deze gevalstudie is slechts bedoeld als illustratie en is dus niet tot in detail uitgewerkt. Ook voor de praktijkinformatie betreffende deze gevalstudie kon ik beroep doen op de bereidwillige medewerking van de heer M. Linten van het bedrijf SKF te Tongeren.

1.3.4 Afbakening onderzoeksdomein

Binnen deze thesis hanteren we een pragmatische visie op agenttechnologie. Het doel is immers om een analyse uit te voeren om zo een beeld te schetsen van de bruikbaarheid van agenttechnologie binnen logistieke netwerken. Deze thesis spitst zich dus niet toe op agenten binnen de wereld van Artificial Intelligence (AI) waar op een abstractere manier wordt gedacht over agenten.

Verder wil ik enkel het gebruik van softwareagenten beschouwen. Hoewel ik uit het gesprek met prof. dr. Holvoet en dr. ir. Roos heb kunnen leren dat het gebruik van agenttechnologie in de wereld van de robotica ook erg interessant is, wordt deze tak van de agentwereld hier niet in beschouwing genomen. Verder wordt ook de toepassing van agenten in de mobiele wereld niet in detail besproken daar dit onderwerp een thesisonderwerp op zich zou kunnen vormen. Hoewel mobiele toepassingen met behulp van PDAs, GSMs en laptops erg nuttig kunnen zijn, kunnen we stellen dat een vaste basisinfrastructuur een hogere prioriteit verdient daar het onderzoek naar agenttoepassingen zelf nog niet de maturiteitsfase bereikt heeft.

Tot slot kunnen we stellen dat tijdens de studie van de realisaties in de agentwereld de hoogste prioriteit gegeven wordt aan de operationaliteit van agenttoepassingen. Hoewel de veiligheid van een agenttoepassing erg belangrijk is, ga ik er van uit dat de aandacht voor de veiligheid niet de hoogste prioriteit verdient als we willen analyseren hoe agenttechnologie de logistieke sector kan ondersteunen. Naar mijn bescheiden mening is een studie van de veiligheidsoverwegingen en –vereisten van multi-agentsystemen een complex en uitgebreid domein dat niet verenigbaar is met dit exploratief onderzoek van agenttechnologie als ondersteuning voor logistieke netwerken.

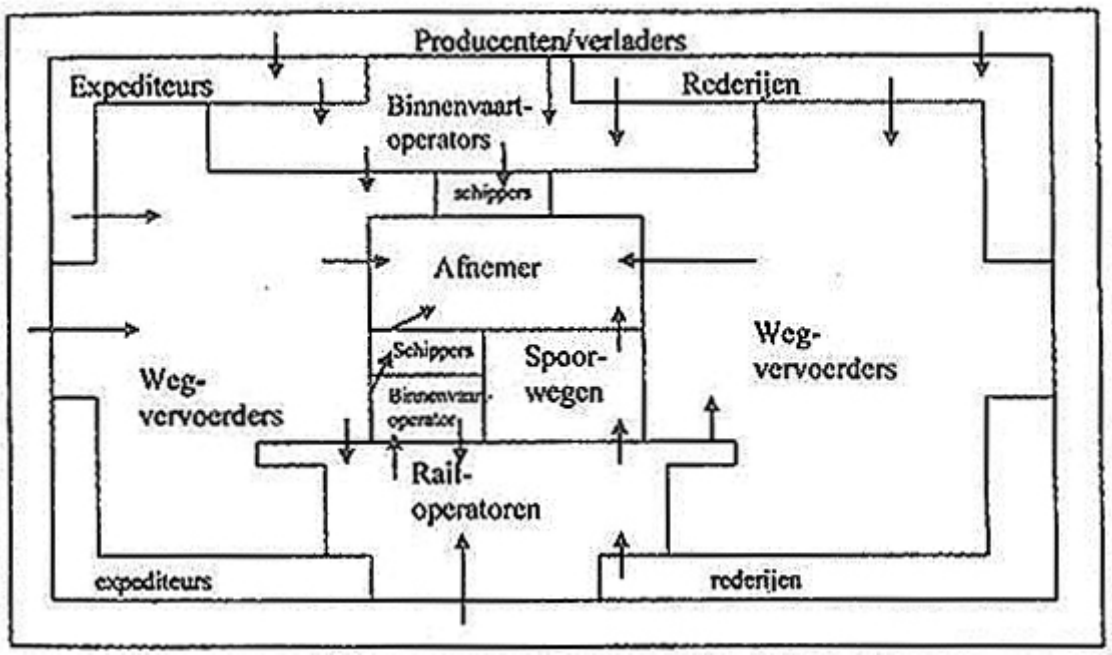
HOOFDSTUK 2: Voorstelling van logistieke netwerken

In de inleiding van deze thesis hebben we de evoluties binnen de logistieke sector al behandeld. We hebben daar aangetoond dat de logistieke sector het laatste decennium complexer geworden is. Om een beter beeld te vormen van de structuur van deze sector en meer bepaald de logistieke netwerken zal in het eerste deel van dit hoofdstuk hier verder op in worden gegaan.

In een volgend onderdeel van dit hoofdstuk zullen we verder ingaan op de huidige manieren van werken binnen de logistieke sector met een focus op de toegepaste IT-systemen. We bespreken kort de manier van werken samen met de voordelen. Achteraf vermelden we ook de tekortkomingen van dergelijke systemen om de dynamische relaties binnen een logistiek netwerk te ondersteunen.

2.1 Actoren binnen een logistiek netwerk

Om eenduidig te beschrijven welke processen in de logistieke keten voorkomen, is gekozen om deze in termen van rollen te beschrijven. Een rol bestaat uit een aantal functies die logischerwijs samenhangen. In de praktijk vervult één actor vaak meerdere rollen. De diverse rollen hebben op verschillende manieren onderlinge relaties. Zo worden bijvoorbeeld goederen en informatie uitgewisseld. Om een beeld te krijgen van de verhoudingen tussen rollen en actoren is in onderstaande figuur aangegeven welke contractuele relaties er bestaan. In bijlage I wordt op elke rol dieper in gegaan met betrekking tot de activiteiten, belangen vanuit interne bedrijfsvoering en hun onderlinge relaties.



Figuur 1: Actoren en relaties in de multimodale keten (Coebergh, 1999)

2.2 Relaties binnen een logistiek netwerk

Actoren vervullen in een logistiek netwerk bepaalde functies afhankelijk van de beslissingen die de organisator van de keten neemt. Deze functies zijn belangrijk om de onderliggende relaties tussen de actoren beter in te zien. Tabel 1 toont een overzicht van de functies in een multimodale keten en geeft een overzicht welke actoren welke functies mogelijkerwijs kunnen opnemen.

In de tabel heeft de marketing- en verkoopfunctie enkel betrekking op het verspreiden van informatie en de stimulatie van de vraag en de creatie van een transactie-uitwisseling. De 'value added services' zijn activiteiten die een actor aanbiedt naast zijn logistieke diensten. Voorbeelden hiervan zijn verpakken en assembleren. Coebergh (1999) stelt dat de belangrijkste functies betrekking hebben op het voorzien van het juiste materiaal op de juiste tijd. De beschikbaarheid van essentiële structuren om de goederen in te laden in combinatie met de logistieke diensten die moeten verleend worden en de kritieke parameter tijd, maken een keten tot een erg complex systeem. Het is zo dat de onderlinge actoren in dergelijke keten erg afhankelijk zijn van elkaar en sterk moeten samenwerken om aan de hoge eisen van de verlader te voldoen.

Tabel 1: Overzicht van de functies opgenomen door de actoren in de transportketen (eigen verwerking van Macharis 2004)

	Infrastructuur voorzien. (publiek en privaat)	Marketing/Verkoop	Materiaal voorzien (containers, trailers, wisselaadbakken…)	Stuffing (vullen containers)	Voor- en natransport (naar en van terminal)	Line Haul (fysieke transport binnenland)	Terminal Operaties (laden en lossen van transportmodi)	Stripping (leegmaken containers)	Voorraadbeheer (van lege containers)	Coördinatie en contracten	Materiaal Management (Transportmodi zelf, infrastructuur)	Facturatie	Douane	Value added services
Overheid	x												x	
Verlader				x	x			x	x				x	
Rederij			x	x				x	x	x	x	x	x	
Expediteurs				x				x	x	x		x		x
Wegtransporteurs		x	x	x	x	x		x		x	x	x		x
Intermodale spoororganisator	x	x	x			x				x	x	x		x
Binnenschipper			x			x					x			
Wegtransporteur voor- natransport				x	x			x			x			
Binnenvaart terminaloperators	x	x	x		x		x		x	x	x	x	x	x
Spoor terminal	x						x		x		x		x	x
Maritieme terminal				x			x	x			x		x	x

Uit voorgaande schets blijkt duidelijk dat een logistiek netwerk een complex organisme is dat bestaat uit vele actoren met elk hun eigen doelstelling. Er is een hoeveelheid aan functies die vervuld moeten worden, waarvoor meerdere actoren zich als kandidaat opstellen om deze te voltooien. Toch is het noodzakelijk dat alle actoren een goede

onderlinge samenwerking aan de dag leggen teneinde de wensen van de verlader zo goed mogelijk te kunnen invullen. Het is dus zo dat de mate van coöperatie, samenwerking en de mogelijkheid om constant te communiceren en te negotiëren van belang is voor de optimalisatie van de totale transportketen en de werking van elke individuele speler in een logistiek netwerk.

2.3 Ondersteuning logistieke processen door IT

De uitwisseling van informatie en coöperatie van verschillende partijen is gerechtvaardigd door het feit dat alle partijen voor een gemeenschappelijk doel samenwerken, namelijk het zo efficiënt en effectief organiseren van de supply chain. Dit fenomeen kan men ook omschrijven als het concept van de 'extended enterprise' (Ranganathan et al, 2004). Het concept van de 'extended enterprise' gaat uit van coöperatie, integratie en nauwe samenwerking tussen bedrijven en hun supply chain partners.

Toch blijkt uit verschillende studies van de Belgische logistieke sector dat het concept van de 'extended enterprise' nog niet veelvuldig toegepast wordt. Zo besluit Macharis (2004) na haar onderzoek dat de mate van integratie binnen de logistieke sector erg laag is. Een logisch gevolg van voorgaande vaststellingen is dat de fragmentatie van data hoog is. Deze kenmerken brengen het negatieve effect met zich mee dat er een erg inefficiënte stroom van informatie is. Data moet door medewerkers uit bedrijfssystemen gehaald worden en doorgestuurd naar medewerkers van de partners. Deze dienen dan de gegevens weer manueel in hun eigen systeem in te voeren. Deze handelingen leggen een hypotheek op de snelheid waarmee informatie beschikbaar wordt. Door deze vertraging is een dynamische en flexibele integratie van de supply chain praktisch onmogelijk. Verder zijn de kosten van de communicatie en arbeid die met die administratie gepaard gaan enorm en in feite overbodig. (Bernaer et al., 2006; Macharis, 2004)

Voor logistieke spelers is het dus niet alleen van belang de interne informatievoorzieningen te ondersteunen maar is het ook de bedoeling dat zij een integratie van de supply chain nastreven. "Het gebruik van innovatieve ICT-toepassingen door logistieke dienstverleners is absoluut noodzakelijk om aan de toenemende eisen van

verladers tegemoet te komen." (Nederland Logistiek Netwerkland, 2006) Van Toor (2004) stelt onder andere dat verdere ICT-ontwikkelingen hopelijk leiden tot volgende trends in de supply chain:

- Minimaliseren van reactietijden in de richting van klanten door het verkorten van de levertijd;
- Realiseren van een integrale besturing van verschillende bedrijfsprocessen;
- Opzetten van de afstemming en/of integratie van opeenvolgende schakels in de keten op operationeel niveau;
- Ontwerpen en implementeren van samenwerkingsverbanden tussen diverse actoren;
- Elimineren of reduceren van voorraden met behulp van innovatieve concepten.

De intense coördinatie die binnen een logistiek netwerk nodig is om bovenstaande doelstellingen te bereiken heeft implicaties voor de juistheid, accuraatheid en snelheid waarmee informatie voorhanden moet zijn. Het is net daarom dat informatie- en communicatietechnologie (IT) een belangrijke katalysator is voor logistieke spelers. Gezien de grote belangen die logistieke spelers hebben met het introduceren van ICT om de supply chain te integreren, staat de logistieke sector bekend om zijn eerder conservatief karakter betreffende vernieuwende technologieën. Een bijkomende moeilijkheid is het complexe karakter dat gepaard gaat met logistieke processen. In vergelijking met andere bedrijfsprocessen zijn er, zoals aangetoond, vele externe partners betrokken bij dergelijk proces. Deze betrokkenheid van externe partners maakt het implementeren en het doordringen van technologie moeilijker dan wanneer het gaat over IT die interne processen dient te ondersteunen. Verschillende partijen dienen samen te werken om een bepaald systeem te implementeren om zo op elkaar afgestemd te geraken.

In dit onderdeel zullen we verder ingaan op de huidige manieren van werken binnen de logistieke sector met een focus op de toegepaste IT-systemen. De nadruk zal hier vooral liggen op de integratie van de verschillende partners in een supply chain omdat dit eigenlijk de kern is van de integratie van logistieke partners. Men kan een onderscheid maken tussen de IT-ondersteuning voor de planning binnen een supply chain en de IT-ondersteuning voor het uitvoeren van de supply chain. De eerste groep bevat vooral de systemen voor planning, analyse en andere beslissingsondersteunende activiteiten.

Hieronder ressorteren toepassingen als ERP-systemen en planningsoftware voor de supply chain. De tweede groep bevat de systemen voor het beheren van de data en de communicatie tussen de partners. Hierbij gaan we dieper in op concepten als EDI, XML en het Internet. We bespreken kort de manier van werken samen met de voordelen die deze systemen kunnen bieden. Achteraf vermelden we ook de tekortkomingen van dergelijke systemen.

2.3.1 ERP-systemen

Heden ten dage heeft bijna elk bedrijf een ERP (Enterprise Resource Planning) systeem. Laudon en Laudon (2005) omschrijven een ERP-systeem als volgt: "Enterprise systems integrate the key business processes of an entire firm into a single software system that enables information to flow seamlessly throughout the organization." Net als uit de definitie blijkt ligt de nadruk van ERP-systemen op de afstemming van de functionele gebieden binnen een bedrijf. ERP-systemen richten zich dan ook eerder op transactieverwerkende activiteiten binnen een onderneming. Op die manier zijn ze hoegenaamd niet geschikt om inter-organisationele processen te ondersteunen. Deze processen over verschillende bedrijfsgrenzen heen zijn immers moeilijker controleerbaar en aan meerdere invloeden onderhevig dan processen die zich strikt binnen de bedrijfsgrenzen afspelen. Het is immers zo dat bij commerciële transacties twee keer data geverifieerd en in de centrale database gezet moet worden, namelijk eenmaal bij de leverancier en eenmaal bij de klant. Dit proces is erg tijdrovend en kostelijk. (Markus, 2000: geciteerd in Nassen, 2006; Wortmann en Szirbik, 2001)

De natuur van de eerste soort van ERP-systemen vereiste een weinig flexibele structuur voor data en processen. Deze eerder inflexibele werking van ERP-systemen wordt door Markus (2000: geciteerd in Nassen, 2006) als een gebrek aan ondersteuning voor het 'extended enterprise' concept gezien. Bovendien viel een gesloten en terughoudend karakter van deze systemen op omdat de commerciële organisaties hun ERP-pakket wilden beschermen en de concurrenten wilden afschermen.

Voorgaande gebreken aan mogelijkheden om supply chain processen te ondersteunen, zorgden voor alternatieve applicaties naast ERP. Door de grote groei van SCM werden ERP-producenten aangezet meer en meer open architecturen te bedenken en

mogelijkheden in te bouwen om SCM te ondersteunen in hun systemen (Nassen, 2006). Enkele ERP-leveranciers realiseerden open interfaces zodat informatie eenvoudig tussen organisaties uitgewisseld kon worden (Verduijn et al., 2003). Verder werd de logge ERP structuur vervangen door een modulair systeem waarin men 'op maat van de klant' modules kon inpassen waaronder dus ook supply chain modules. Bovendien gingen ERP producenten zichzelf ook meer toeleggen op processen die zich ook buiten de ondernemingsgrenzen afspelen zoals SCM en CRM (Nassen, 2006).

Deze vlaag van vernieuwing zorgde ook voor de introductie van een nieuwe term namelijk ERPII. Gartner Group definieert ERPII als: " a business strategy and a set of industry-domainspecific applications that build customer and shareholder value by enabling and optimizing enterprise and inter-enterprise, collaborative operational and financial processes." (Wood et al., 2000: geciteerd door Nassen, 2006)

Voor bedrijfsprocessen waarbij relaties met externe partners moeten ondersteund worden, hebben bedrijven dus enerzijds de keuze uit een complete implementatie van een ERP pakket inclusief de modules voor SCM. Anderzijds kunnen ze er ook voor kiezen om het ERP pakket als basis te gebruiken waarmee men dan SCM pakketten en dergelijke mee kan integreren. In volgende alinea bespreken we kort de mogelijkheden van ERP pakketten in organisatiewijde integraties.

De omzetcijfers van commerciële ERP-systemen bereikten de laatste jaren hun hoogtepunt omdat de groep ERP-leveranciers, zoals SAP maar ook Oracle, Sage Group etc…, bijna in elk bedrijf een ERP-pakket geïmplementeerd heeft. Hoewel zij claimen om een open interface te bezitten om informatie tussen ondernemingen te laten 'stromen' is het zo dat dit tussen de aangehaalde ERP-pakketten helemaal niet het geval is. Vaak komen hier extra datamigraties of vertaling aan te pas die natuurlijk de kosten de hoogte indrijven (Verduijn et al., 2003). Ranganathan et al. (2004) hebben het in hun paper ook over de integratie van supply chains via IT. Zij zien ERP alleen als een eerste stap van integratie, namelijk de integratie van interne processen met de logistieke functie. Deze fase zouden we dan als een voorbereidende fase kunnen zien om via nieuwe SCM en web-technologie ook applicaties uit te bouwen voor bedrijfsprocessen over bedrijfsgrenzen heen. Zij denken dus net als Verduijn dat commerciële ERP pakketten in het algemeen nooit zullen zorgen voor dergelijke integratie over de supply chain heen.

2.3.2 EDI/XML in de supply chain

SCM-pakketten proberen de communicatie- en gegevensuitwisseling in een supply chain te ondersteunen door middel van Electronic Data Interchange (EDI). EDI geeft bedrijven de mogelijkheid elektronische standaardberichten uit te wisselen tussen incompatibele systemen. Toch blijkt uit onderzoeken dat de introductie van EDI niet erg veel succes heeft gekend in de vele bedrijven over de hele wereld. De reden voor de lage implementatiegraad van EDI is de moeilijkheid waarmee het geïntegreerd kan worden met bestaande systemen binnen de ondernemingen. (Verduijn et al., 2003)

De integratie van EDI is immers een kostelijke zaak zodat veel, vooral kleinere, bedrijven niet bereid waren een systeem op basis van EDI te implementeren. Dit gegeven maakte de drempel om toe te treden tot een EDI systeem nog hoger mits de voordelen van EDI door de supply chain lager zijn als niet alle partners EDI gebruiken. Kritische factor voor het succesvol zijn van dergelijk systeem is de implementatie over de volledige supply chain. Vaak hervalt men dan ook in een machtsspel waarbij kleinere bedrijven door grote partners verplicht worden dergelijke systemen te implementeren. Deze traditionele systemen leidden initieel tot hoge 'lock-in' kosten en dus ook hoge overstapkosten zodat partners gebonden waren aan mekaar (Ranganathan et al., 2004). Deze bevindingen werden over de volle lijn bekrachtigd door het interview dat werd afgenomen met de heer Linten van SKF.

Skonnard (2002: geciteerd door Depaire, 2003) vermeldt dat XML een revolutie betekende in het ontwikkelen van gedistribueerde toepassingen. XML is een semantische markeertaal. Daarmee bedoelen we dat het zich richt op de structuur en de semantische betekenis van een document. Op die manier is XML een formaat waarmee op een flexibele en effectieve manier data uitgewisseld kunnen worden tussen verschillende applicaties. XML bestaat uit een set labels die tot een minimum beperkt zijn maar die uitbreidbaar zijn door de gebruikers. De gebruiker kan zijn eigen labels definiëren die hij kan vullen met inhoud. Op deze manier kan men via XML labels en waardes uitdrukken en kan het zinvol zijn om berichten te coderen. Zogenoemde 'tags' worden rond de inhoud die men wil uitwisselen geplaatst. Via deze 'tags' kan men afleiden wat de inhoud van die variabele is. Op deze manier is XML makkelijker te integreren met bestaande systemen omdat ze beter te linken zijn met bestaande databasestructuren.

Via EDI en XML wordt het dus mogelijk informatie uit te wisselen tussen voorheen incompatibele systemen. Toch is de integratie van dergelijke systemen vooral in het geval van EDI moeilijk en kostelijk. We onthouden ook dat we steeds moeten voorzien in een gemeenschappelijke standaard of protocol waarin een vaste ontologie of 'woordenboek' van termen en variabelen bestaat. Bij XML is het mogelijk om deze ontologie om te zetten via het definiëren van nieuwe XML-tags. XML is hierdoor veel flexibeler in het gebruik dan EDI waar men binnen een vaste structuur dient te werken. We kunnen besluiten dat beide technologieën kunnen zorgen voor een daling van de fragmentering aan data. Op deze manier wordt manuele input in de afzonderlijke systemen beperkt.

2.3.3 Advanced Planning Systems

In de logistieke sector zijn er bepaalde softwaretoepassingen speciaal ontwikkeld om een gedetailleerde planning uit te tekenen als ondersteuning voor supply chain processen. Traditioneel bestaan dergelijke Advanced Planning Systems (APS) uit verschillende modules die elk een aspect van een dergelijke planning voor hun rekening nemen. Op deze manier kan men zowel op korte-, middellange- als langetermijn diverse supply chain processen plannen binnen een onderneming. (Laudon en Laudon, 2005)

Toch is de werking van een APS niet erg effectief wanneer het inter-organisationele processen dient te plannen. Vaak dient bij een Supply Chain proces informatie van meerdere logistieke spelers opgenomen te worden en het is net hier dat het schoentje wringt. Zoals reeds vermeld werd, is de logistieke sector gekenmerkt door incompatibele systemen, fragmentering van data, verschillende protocollen om informatie uit te wisselen, andere definities en betekenissen van bepaalde variabelen etc. Het is dan vaak ook niet mogelijk om op een efficiënte manier een planning op te maken met behulp van deze gespecialiseerde software. (Verduijn et al., 2003)

Verder is het zo dat plannen en realiteit zelden samenvallen. De realiteit zal er altijd anders uitzien dan de plannen die op voorhand werden opgesteld. Door data te consolideren en om te zetten naar gebruiksvriendelijke data voor de planningsmodules, zou men er in slagen een kwalitatief plan te ontwikkelen. Toch is het niet mogelijk om met alle onvoorziene omstandigheden rekening te houden binnen dit plan. Op dit gebied

schieten deze applicaties dus tekort. We willen echter niet stellen dat het plannen van activiteiten niet aan de orde is. Het is immers net dit proces dat richting geeft aan inspanningen en ervoor zorgt dat de realiteit zo optimaal mogelijk uitgevoerd kan worden. Toch moeten we ermee rekening houden dat planningsmodules niet kunnen omgaan met het dynamisme en de flexibiliteit die nodig is om de uitvoering van de keten te ondersteunen.

2.3.4 Internet en E-toepassingen

Het ontstaan van Internet heeft voor vele opportuniteiten gezorgd binnen de zakenwereld. Toch is het Internet slechts een medium waarmee bepaalde oplossingen bewerkstelligd kunnen worden en geen oplossing op zich.

Verduijn et al. (2003) spreekt over een mooie Internet toepassing namelijk e-markets. E-markets bieden een platform waar kopers en verkopers aan elkaar gelinkt kunnen worden en transacties uitvoeren. Via deze web-technologie en B2B-toepassingen is het mogelijk om op een efficiëntere en goedkopere manier informatie te verwerken en door de supply chain te laten stromen. Zoals aangetoond in het voorbeeld met e-markets is internettechnologie hier slechts een medium dat wordt gebruikt door de menselijke agenten van het bedrijf. Overigens verloopt de integratie met het ERP-pakket ook niet altijd van een leien dakje (Brunn et al., 2002). De medewerkers moeten actief deelnemen aan deze transacties, data verwerken en continu de markt en de noden van het bedrijf bestuderen.

2.3.5 Kritische succesfactoren bij SCM toepassingen

De mate van succes bij webgebaseerde toepassingen en EDI voor supply chain management is gerelateerd aan een aantal factoren. In hun paper testen Ranganathan et al. (2004) een tiental hypothesen die de mate van toepassing en gebruik van dergelijke technologieën proberen te verklaren in zowel het interne als externe bedrijfslandschap. Zijzelf noemen dit de interne assimilatie en externe diffusie van web- en EDI-technologie.

In dit werk wordt formeel bekrachtigd dat de relatie tussen het gebruiken van Webtechnologie binnen de bedrijfsgrenzen en het gebruik ervan voor de processen buiten

de bedrijfsgrenzen nauw verbonden zijn. De verspreiding van technologie over bedrijfsgrenzen heen (diffusie) wordt in grote mate bepaald door de onderlinge afhankelijkheid tussen verschillende partners. Verder vertoont de mate van toewijding aan een bepaalde technologie een directe relatie met de voordelen die deze technologie oplevert. Dit kan verklaard worden door het feit dat men bepaalde vaardigheden en kennis van de toegepaste technologie beter onder de knie begint te krijgen en men een soort van organisationeel leereffect creëert.

De vaststelling in voorgaande alinea betekent dus dat de voordelen van een EDI systeem bijvoorbeeld groter zullen zijn als ook de diffusie van deze technologie onder de partners groter is. Verder betekent de nauwe relatie tussen de diffusie en assimilatie van een technologie dat wanneer de diffusie tussen de partners groter is men ook een hogere assimilatie gaat hebben van deze technologie. Een bedrijf gaat zich dus meer focussen op de technologie die ook door zijn partners gebruikt wordt omdat hij daar intern zijn processen ook op wil afstemmen. Echter, zoals we gezien hebben zijn de relaties binnen een logistiek netwerk talrijk, complex en dynamisch. Een volledige diffusie van web- en EDI-technologie die volledig op elkaar afgestemd zijn in dit logistiek netwerk is in deze context praktisch niet haalbaar.

Een ander belangrijke relatie die bekrachtigd werd is het fenomeen dat een centraal geleide implementatie en structuur van een technologie zal zorgen voor een lagere graad van assimilatie door bedrijven dan wanneer deze implementatie decentraal is. Deze significante hypothese helpt ons te verklaren waarom EDI en SCM systemen niet erg succesvol zijn in de logistieke sector en de mate van datafragmentatie nog steeds zo hoog ligt. Met de aangehaalde systemen is het immers vaak zo dat de meer machtige speler zijn wensen over het systeem dat in de supply chain gebruikt dient te worden, oplegt aan zijn partners. Vaak slaagt deze machtigste speler in deze handeling doordat de andere partners inzien dat dergelijk systeem zorgt voor meer afhankelijkheid tussen beide bedrijven wat reeds in de bovengaande alinea uitgelegd werd. Deze praktijk gebeurt vooral bij het toepassen van EDI waarbij de minder machtige bedrijven noodgedwongen de wensen van de meer machtige spelers moeten volgen. Toch is dergelijke manier van werken niet bevorderlijk voor de werkelijke toewijding of assimilatie die bedrijven hebben voor een bepaalde technologie.

2.4 Besluit huidige IT ondersteuning

Bij alle systemen hebben we gezien dat deze een positieve bijdrage kunnen leveren aan de efficiëntie en effectiviteit van een onderneming. Hoewel ERP in de meeste gevallen enkel geschikt is als systeem voor de interne processen, is het een bouwsteen waarop SCM toepassingen gebouwd kunnen worden. Wanneer we kijken naar de integratie van deze systemen zien we dat het probleem van integratie met bestaande systemen vaak de kop opsteekt. Verder bieden deze toepassingen te weinig flexibiliteit en dynamiek om de uitvoering van een supply chain te ondersteunen.

Een ander belangrijk probleem is dat men met verschillende bedrijven op dezelfde golflengte moet zitten en de onderlinge afspraken moet respecteren als men een integratie wil bekomen. Deze samenwerking en participatie is immers kritisch voor het succes en de mate van gebruik van de technologie. Toch moeten bij de onderhandelingen in de huidige logistieke sector de meeste bedrijven vaak te grote toegevingen doen aan een machtigere speler en hun interne structuur drastisch aanpassen.

HOOFDSTUK 3: Bespreking agenttechnologie

Doordat het paradigma van de agent een erg robuuste en bijna natuurlijke manier is om applicaties en systemen voor te stellen zijn vele onderzoekers enthousiast over het potentieel dat agenttechnologie bezit. Zogenaamde agentsystemen zijn computersystemen die abstract voor te stellen zijn in de vorm van agenten en waarin we enige vorm kunnen herkennen van agenttechnologie, zoals beschreven in de inleiding van deze eindverhandeling. Dit hoofdstuk wil een niet exhaustieve samenvatting zijn van opvattingen in de bestudeerde literatuur over het denken, de mogelijkheden en de kenmerken van agenten en agentgebaseerde systemen.

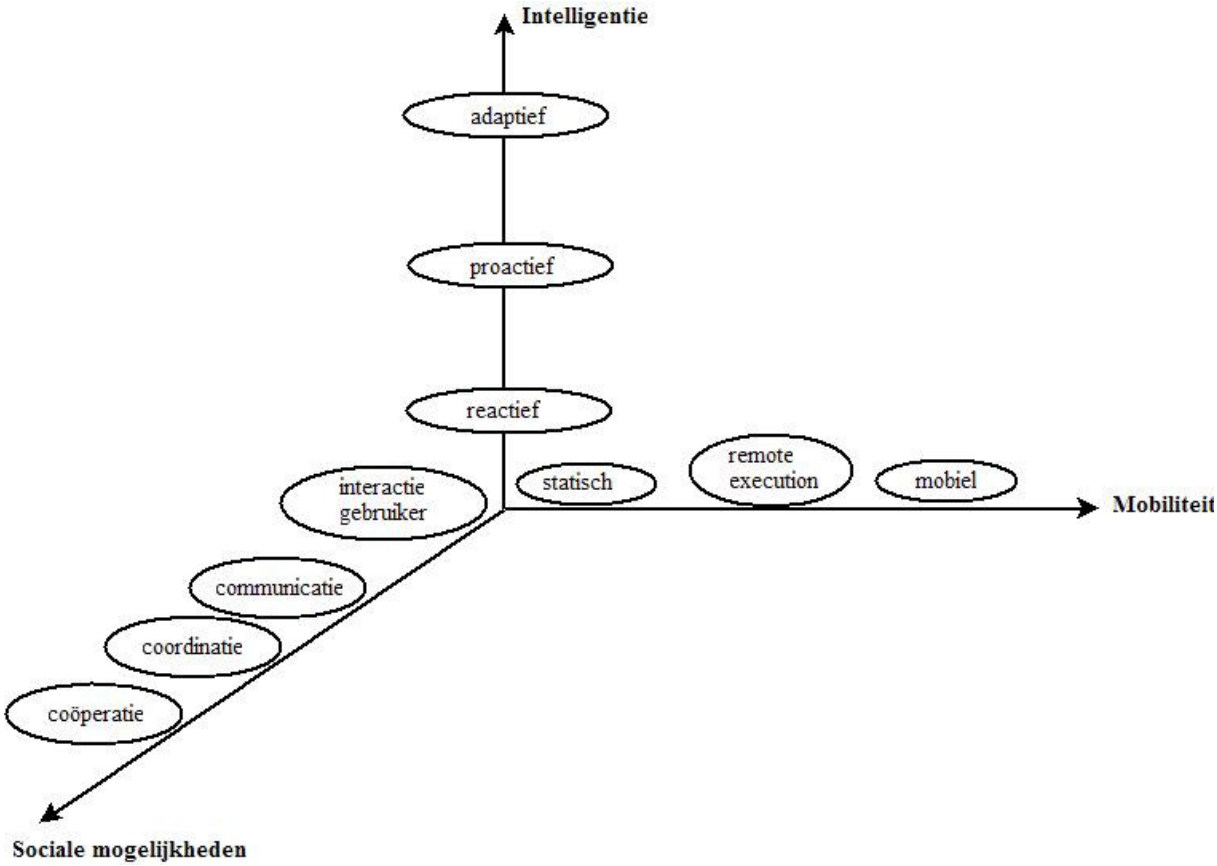
3.1 Classificatie van agenten op basis van eigenschappen

In hoofdstuk 1 is reeds een deeltje van de sluier opgelicht over de eigenschappen die agenten bezitten. We vermeldden dat de belangrijkste kwaliteiten van een agent de autonomie, het doelgericht gedrag en het sociale karakter zijn. Zoals we gezegd hebben uit het doelgericht gedrag zich via reactief en proactief handelen. In dit gedeelte wordt een meer uitgebreid beeld gegeven over de opvattingen binnen de wetenschappelijke wereld betreffende de kwaliteiten die agenten dienen te bezitten. Deze complexere visie op agenten wordt ook wel 'a stronger notion of agency' genoemd (Jennings en Wooldridge, 1994).

Vele auteurs wagen zich aan een classificatie van een agent op basis van de eigenschappen die ze bezitten. Bij de meeste auteurs vinden we een schaal terug in drie dimensies zijnde 'mobiliteit', 'sociale eigenschappen' en 'intelligentie'. Buiten deze eigenschappen is het belangrijk dat we ons realiseren dat de agent verplicht een zekere mate van autonomie bezit. (Gelati, 2004; Jennings et al., 1998; Gilbert et al., 1995:geciteerd door Vulkan, 1999) Andere auteurs wagen zich niet aan deze moeilijke

oefening omdat zij menen dat het niet mogelijk is een allesomvattende voorstelling te maken van de kwaliteiten die een agent kan bezitten. Volgens hen zijn de eigenschappen die agenten bezitten enkel voor te stellen in een multi-dimensionele ruimte. Daarom beperken zij zich tot een opsomming en definiëring van agenten met hun verschillende kwaliteiten (Nwana, 1996).

Gezien de kracht en het versneld inzicht die een voorstelling mogelijkerwijs kan bieden, wil ik toch even een voorstelling te geven van de kwaliteiten van agenten zoals terug te vinden in de literatuur. Zo verkiezen Vulkan (1999) en Gelati (2004) om de typologie van agenten op een driedimensionale schaal voor te stellen. Ik koos ervoor om een soort van synthese te maken van de opvattingen binnen de doorgenomen literatuur. Op deze manier plaatste ik op de X en Y-as respectievelijk de graad van mobiliteit en intelligentie en op de Z-as de graad van sociale interactie. Deze figuur wordt uitgebreid besproken in de punten 3.1.1 tot en met 3.1.3.



Figuur 2: Voorstelling agentkwaliteiten

Zoals reeds vermeld waagt Nwana (1996) zich niet aan het opstellen van dergelijk assenstelsel. Zij ziet eerder een agent als een software applicatie die verplicht meerdere van volgende eigenschappen bezit: 'autonomy', 'cooperation' and 'learn'. Toch onderscheiden agenten die twee of meerdere eigenschappen bezitten zich van elkaar door een reeks andere eigenschappen. Nwana haalt de aard van mobiliteit en de manier waarop agenten beslissingen nemen aan. De gelijkenissen met bovenstaand assenstelsel zijn legio betreffende deze eigenschappen.

Bij Jennings en Wooldridge (1994) vinden we buiten de reeds vermelde eigenschappen ook nog oprechtheid, welwillendheid en rationaliteit terug. Tot slot zijn er een hele reeks secundaire eigenschappen waardoor agenten zich onderscheiden zoals de ingesteldheid van de agent (afwijzend of aanvaardend), de veelzijdigheid van een agent (mogelijkheid om meerdere taken te vervullen bijvoorbeeld), de continuïteit van de agent... Deze secundaire kenmerken breiden constant uit met nieuwe menselijke kenmerken en zouden een agentvoorstelling multidimensioneel en erg complex maken.

Verder is een ander belangrijk aspect van een agent natuurlijk de rol die hij toebedeeld krijgt. Agenten kunnen 'stand-alone' applicaties zijn of deel uitmaken van een multi-agentsysteem waar agenten instaan voor specifieke taken. Hier zullen we later in dit hoofdstuk verder op ingaan.

In de beschrijvende literatuur over agenten zijn er buiten de vele verschillende opvattingen toch enkele constante vaststellingen af te leiden. Een agent is geen waardige agent als zijn mogelijkheden beperkt zijn tot één domein. Vulkan (1999) vermeldt terecht het voorbeeld van een expertsysteem dat enkel een bepaalde graad van intelligentie bezit en dus niet als agent beschouwd kan worden omdat er niet gescoord wordt op een ander domein. Verder is autonomie vaak een vereiste voorwaarde om te kunnen spreken over een agent. Samen met deze autonomie is de manier van berichten en opdrachten uitwisselen binnen een agentsyteem via agentcommunicatie tekenend voor agenttechnologie. Tot slot kan ik vermelden dat agenten een erg heterogene verzameling aan softwareobjecten is. Daardoor is er vaak discussie mogelijk of een softwareobject al dan niet een agent is (Anumba et al., 2001: geciteerd door Verduijn et al., 2003)

3.1.1 Sociale vaardigheden bij agenten

Een belangrijk kenmerk van een agent is de mogelijkheid om interactie aan te gaan met andere agenten. Algemeen kunnen we deze interacties onderverdelen in een aantal verschillende soorten interacties. Zo halen Jennings et al. (1998) aan dat de meest voorkomende vormen van interacties tussen agenten coöperatie, coördinatie en negotiatie zijn. Andere auteurs vermelden volgende volgorde van interacties in stijgende complexiteit: communicatie, coördinatie, coöperatie en collaboratie (Wortmann en Szirbik, 2001).

Hoewel de twee beweringen in voorgaande alinea nauw bij elkaar aansluiten, bespreken we de mogelijke interacties tussen softwarecomponenten volgens het vierlagige model dat ontwikkeld is door Diets en Widdershoven (1992: geciteerd door Wortmann en Szirbik, 2001). Volgens dit model is communicatie de eenvoudigste vorm van interactie. Deze vorm van interactie kan gebruikt worden om aan data-integratie[I] te doen. Dit wil zoveel zeggen als data van het ene systeem overbrengen naar het andere via een communicatietaal met een voorgedefinieerde syntax[II]. Bovendien wordt de gebruikte semantiek[III] expliciet afgesproken en gedeeld door beide partijen. Aangezien deze vorm van interactie zich op een erg laag niveau situeert en eigenlijk eigen is aan alle softwarecomponenten, is het aan te raden dat een agent een complexere vorm van interactie ondersteunt. (Wortmann en Szirbik, 2001)

De tweede vorm van interactie wordt gedefinieerd als coördinatie. Jennings et al. (1998) definiëren deze interactie waarlangs men een procesintegratie kan bekomen als volgt: "organising problem solving activity so that harmful interactions are avoided or beneficial interactions are exploited". Dit is mogelijk doordat bepaalde processen geïdentificeerd worden. Deze identificatie wordt gedeeld door beide partijen zodat men gericht procesinformatie kan delen. Beide systemen blijven dus hun private semantiek behouden. Deze vrijheid aan semantiek is mogelijk doordat de verzonden berichten meta-data over de inhoud bevatten. Op deze manier kan er een vertaling gebeuren van

---

[I] Deze vorm van integratie wordt gerealiseerd door het gebruik van EDI en XML zoals besproken in hoofdstuk 2.
[II] De syntax van een bericht zijn de richtlijnen betreffende de structuur van de symbolen m.a.w. de constructie van een bericht. (Singh, 1998)
[III] De semantiek is de wetenschap die zich bezig houdt met de betekenis van taalkundige constructies. (Singh, 1998)

de waardes in het bericht naar de eigen semantische waarden in het systeem. (Wortmann en Szirbik, 2001)

De derde en meest complexe vorm van interactie die mogelijkerwijs geïmplementeerd kan worden door agenten is coöperatie. Bij coöperatie kunnen agenten gemeenschappelijke win-win situaties ontdekken en door hun mogelijkheden om te negotiëren tot een bepaalde beslissing komen. Hoewel Jennings et al. (1998) zowel coöperatie als negotiatie als een vorm van interactie bestempelen is dit in het model van Diets en Widdershoven (1992: geciteerd door Wortmann en Szirbik, 2001) niet het geval. Het is immers zo dat deze twee laatste auteurs de mogelijkheid van agenten om te negotiëren zien als een onderdeel van het complexe interactiepatroon dat zij als coöperatie bestempelen. Coöperatie kunnen we zien als het samenwerken naar een gemeenschappelijk doel terwijl negotiatie het tot een akkoord komen over bepaalde items is.

De meeste complexe vorm van interactie is volgens Diets en Widdershoven collaboratie. Deze vorm van interactie gaat nog verder dan coöperatie en zij vermelden dat dergelijke vorm van interactie bijna onmogelijk te realiseren is bij softwareagenten. Daarom gaan wij ook niet verder in op deze vorm van interacties.

3.1.2 Intelligentie bij agenten

De kern bij agenten is dat ze flexibel kunnen inspelen op veranderende omgevingstoestanden zoals reeds aangehaald in de inleiding. Betreffende intelligentie kunnen agenten een drietal gradaties van flexibiliteit vertonen, namelijk reactiviteit, pro-activiteit en adaptiviteit. Wanneer een agent reactief is zal de programmeur alle mogelijke situaties moeten voorzien en deze implementeren in een ‘rule-base’. Wanneer de agent een impuls opvangt die hij niet kan terugvinden in zijn ‘rule-base’ zal er logischerwijs geen actie gebeuren. Deze vorm van intelligentie behoort eerder tot meer ouderwetse paradigma’s en wordt liefst niet toegepast bij agenten. (Verduijn et al., 2003)

Pro-activiteit bij een agent slaat op de mogelijkheid om omgevingselementen in te schatten en doelgerichte acties te ondernemen. Zij gebruiken hun ‘rule-base’ dus op een

flexibelere manier dan reactieve softwaretoepassingen. Adaptieve agenten zijn in staat om hun eigen handelen bij te stellen doordat zij kunnen leren uit omgevingselementen. Agenten die als het meest intelligent bestempeld worden, kunnen zelfstandig acties ondernemen om zo beter in te spelen op omgevingselementen. Dit kunnen agenten doen door associaties en groeperingsregels toe te passen. Eigenlijk zijn zij in staat om zelfstandig hun rule-base aan te passen aan nieuwe situaties. (Verduijn et al., 2003)

Het is natuurlijk zo dat de intelligentie van een agent slechts zo hoog is als de gesofisticeerdheid van de algoritmes die zij implementeren. Wortzmann en Szirbik (2001) maken dit punt duidelijk: "[…] the level of 'intelligence' of agents is given by the algorithms applied to select one of the alternatives". Verder kunnen we stellen dat het adaptief zijn van agenten eerder van toepassing is in de wereld van de robotica. Deze tak van agenttechnologie, waar we niet verder op ingaan zoals uitgelegd in het onderzoeksopzet, werkt met sensoren om omgevingselementen te percipiëren en op basis hiervan een agent een adaptief gedrag te laten vertonen. (Interview met dr. ir. N. Roos)

3.1.3 Mobiliteit bij agenten

Met mobiliteit in de meest uitgebreide zin van het woord wordt bedoeld dat de agenten de mogelijkheid hebben om zich te verplaatsen op elektronische manier (Hermans, 1996). De status en toestand van een object kan dan gecodeerd worden in een bepaald formaat dat instaat voor de gegevensoverdracht. Naast het andere uiterste namelijk, statische agenten, kan een agent ook via 'remote-execution' aangedreven worden. Gebruikers kunnen dan commando's aan deze agent geven van op een andere machine als degene waar de agent op geïnstalleerd is. In het onderzoeksopzet hebben we aangetoond waarom we niet verder ingaan op deze kwaliteit bij agenten.

3.2 Structuur van een agent

Om bovenstaande functionaliteiten mogelijk te maken dient een agent ontwikkeld te worden volgens een bepaalde opvatting of 'agency'. Een 'agency' kan dienen als een abstracte structuur of denkpatroon voor agenten. In de literatuur is de meest

voorkomende en onderzochte model om een agent te karakteriseren de "Beliefs Desires Intention" (BDI) structuur die ontwikkeld werd door Rao and Georgeff (1995). Via deze opvatting wordt een agent ontwikkeld rond drie datastructuren, namelijk 'beliefs', 'desires' en 'intentions'.

'Beliefs' zijn de percepties van de omgeving van de agent en dus ook de informatie die de agent bezit over zijn omgeving. We kunnen dit omschrijven als het kennismodel van de agent. 'Desires' daarentegen zijn de doelstellingen die een agent wil bereiken. Om zijn doelstellingen te bereiken zal een agent een doelgericht gedrag moeten nastreven dat is afgestemd op de 'beliefs' van een agent. Dit heeft tot gevolg dat de agent bepaalde 'intentions' zal hebben die dan gaan resulteren in acties. Intentions zijn het gevolg van een afstemming van een 'belief' met een 'desire' zodat een agent bereid is om een bepaalde taak of commando uit te voeren. Deze acties worden door agenten uitgevoerd door het uitwisselen van berichten. Deze structuur wordt ook wel het 'mental agency' genoemd.

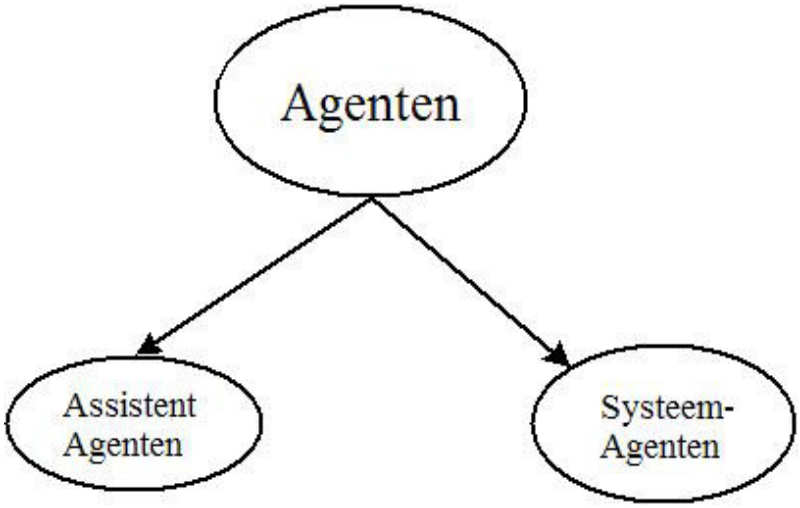
Deze BDI-structuur is reeds in vele applicaties toegepast en vele malen onderzocht. Ook is er tijdens de ontwikkeling van de belangrijkste communicatieprotocols voor agenten, die we later nog zullen bespreken, gebouwd op het BDI-model. Een ander nuttig en veelbelovend model van een agent is het sociaal agentschap, ofwel 'social agency', van Singh (1998; 2000: geciteerd in Chaib-Draa en Dignum, 2002; Jennings en Wooldridge, 1994). Deze opvatting heeft een breder perspectief dan het BDI-model want het biedt aandacht voor de sociale relaties en context binnen een agentsysteem.

3.3 Classificatie van agenten op basis van rol

Agenten kunnen vele rollen vervullen ten einde gebruikers of andere agenten te dienen. In deze rollen maken agenten gebruik van hun verschillende kwaliteiten die eigen zijn aan hun interne structuur. Deze kwaliteiten zijn in voorgaand deel reeds uitgebreid besproken. In deze classificatie maken we een onderscheid tussen agenten die gebruikers ondersteunen en agenten die samenwerken in een multi-agentsysteem.

De eerste soort agenten, 'assistent' agenten genoemd, kunnen onafhankelijk van andere agenten de gebruiker van dienst zijn en zijn taak vervullen. De tweede soort agent

bevindt zich binnen een agentsysteem omdat hij de functionaliteiten van andere agenten nodig heeft om zijn eigen 'gebreken' op te vullen (Luck et al., 2004). Een overzicht van deze twee groepen is terug te vinden in figuur 3. Deze figuur zal dienen als basis om op een dieper niveau de rollen van agenten te bespreken. Het resultaat van de volledige bespreking van de rollen van agenten is terug te vinden in figuur 4 die na de bespreking zelf opgenomen is in deze tekst.



Figuur 3: Classificatie van agenten

De grens tussen assistent- en systeemagenten is eerder vaag mits agenten uit het eerste luik ook kunnen ingepast worden in een multi-agentsysteem. Van de andere kant kunnen in sommige gevallen de agenten verweven in een multi-agentsysteem op onafhankelijke basis belangrijk zijn voor hun gebruiker. Denken we bijvoorbeeld aan een agent die informatie uit een aankoopsysteem haalt ten behoeve van andere agenten maar ook voor een menselijke gebruiker nuttige informatie kan leveren. Hoe deze agenten precies werken wordt hier niet behandeld mits dit niet behoort tot het bestek van deze thesis. Later in dit werk zal natuurlijk wel uitgelegd worden hoe agenten die behoren tot een multi-agentsysteem hun taken kunnen uitvoeren.

3.3.1 Assistent agenten

Agenten die samenwerken met gebruikers en deze bijstaan in het gebruik van software en andere IT infrastructuur noemt men ook wel 'personal assistants' (Papazoglou, 2001). Nwana (1996) heeft het hier over agenten die observeren en monitoren wat de gebruiker

uitvoert op zijn desktop computer. Deze interface agenten zoals zij ze noemt zijn sterk autonoom en hebben een lerend vermogen gebaseerd op 'memory-based' of 'parametric learning'. Deze agenten helpen gebruikers met het uitvoeren van hun taken in software- of Webomgevingen daarmee rekening houdend met het gedrag en feedback van de gebruiker zelf. Op deze manier kunnen zij de gebruiker bijstaan tijdens het werken met applicaties en zelfs vervelende taken overnemen door nabootsing van het gedrag van de gebruiker. Gezien de aard van de taak hoeven deze agenten niet te beschikken over sociale kwaliteiten om te kunnen communiceren met andere agenten. Uitzonderlijk kan dit toch gebeuren omdat een agent informatie vraagt over een bepaald aspect van een programma. Een voorbeeld van dergelijke interfaceagent is WebMate. WebMate is een agent die de gebruiker assisteert bij het zoeken en browsen van gegevens op het Web door het gepast aanbieden van gerelateerde links, interessante pagina's op basis van de reeds uitgevoerde acties van de gebruiker in het verleden. (Chen en Sycara, 1997)

Wellicht de bekendste rol die een agent kan vertolken, is het lokaliseren en het aanbieden van gegevens op het World Wide Web aan een gebruiker. De hoeveelheid aan informatie neemt spectaculair toe en via eenvoudige zoekalgoritmes is het onmogelijk om snel en efficiënt de informatie te vinden die men zoekt. 'Information' agenten zijn in staat om verspreide gegevens van heterogene aard te beheren, detecteren en verzamelen in opdracht van de gebruiker. Papazoglou (2001) haalt voorbeelden aan als Spiders, Lycos of WebCrawlers die voorbeelden zijn van zogenaamde 'spiders'. Zogenaamde 'spiders' zijn indexen van het World Wide Web die in staat zijn de structuur waarin informatie opgesteld is op te slaan en van deze index gebruik te maken wanneer gebruikers hem bepaalde zoekopdrachten laten uitvoeren. Volgens Nwana (1996) is een toepassing pas een echte agent als deze weet hoe hij verschillende webindexen moet ondervragen teneinde nuttige data te verzamelen voor de gebruiker. Etzioni en Weld (1994) presenteerden hun 'softbot' die naast zijn kwaliteiten om informatie te vergaren ook kon omgaan met dubbelzinnigheid en onzekerheid in de zoekopdrachten van gebruikers. De kwaliteiten die een informatieagent moet bezitten zijn heel divers en hangen af van agent tot agent. Ook merk ik nog op dat het complexer landschap van het Web gecombineerd met de vooruitgang op agenttechnologie ervoor gezorgd hebben dat informatieagenten noodgedwongen meer agentgebaseerde systemen worden met als doel het vinden en samenvoegen van heterogene stukken aan data.

Verder is het zo dat een gespecialiseerd en functioneel multi-agentsysteem heden ten dage ook op een dynamische manier over gespecialiseerde informatie moet beschikken. Daarom worden vaak informatieagenten geïmplementeerd in multi-agentsystemen. Deze informatieagent krijgt dan mogelijkerwijs orders van menselijke gebruikers of andere agenten zoals we verderop zullen zien. In deze zin overlappen informatieagenten steeds meer onze volgende klasse van samenwerkende agenten.

3.3.2 Agenten hoofdzakelijk betrokken in multi-agentsystemen

Een agent die instaat voor een welomlijnde functie en hierin gespecialiseerd is, noemt men ook wel een expertagent (Vulkan, 1999). Bij Papazoglou (2001) heet dit soort agenten 'application' agenten. Volgens hem verschaffen deze onder andere toegang tot gespecialiseerde data en kennisdatabanken waarvan zij de kennis moeten beheren en representeren voor andere agenten. Op deze manier kunnen zij 'queries' van andere agenten zo goed mogelijk invullen. Een voorbeeld hiervan is de agent die verantwoordelijk is voor het beheer van de databank van hotels binnen een multi-agentsysteem met als doel het boeken van reizen (van Aart, 2001). Deze agent kan dan aan een agent die de gehele reis coördineert advies geven over de hotelkeuze die het best past binnen de vooropgestelde reis.

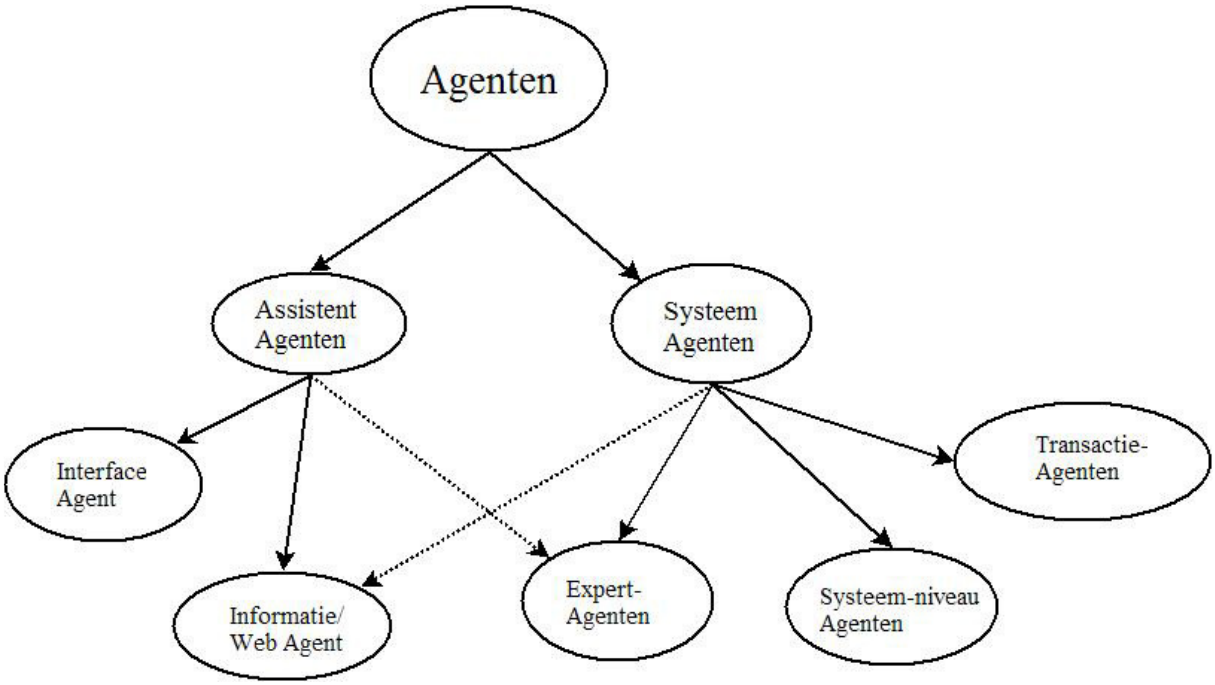
Nwana (1996) houdt het bij 'taskspecific' agenten die hun specifieke domein waarvoor ze verantwoordelijk zijn modelleren in een kennismodel. Verder kunnen zij informatie uit deze specifieke omgeving halen, bezitten zij protocols om hierover te communiceren met andere agenten en bezitten zij strategieën die er op gericht zijn problemen op te lossen. In mijn ogen heeft Papazoglou (2001) het over gespecialiseerde informatieagenten die toegang hebben tot specifieke domeinkennis via databases. Nwana (1996) heeft het duidelijk over een breder en rijker begrip mits deze agent een zeker model en opvatting van de werkelijkheid bezit en op basis hiervan beslissingen kan gaan maken.

Omdat agenten onderling instaan voor het oplossen van complexe problemen in een multi-agent systeem, zijn er ook agenten die op systeemniveau taken moeten uitvoeren. De zogenaamde 'systemlevel support' agenten zijn als een soort administrator voor het multi-agent systeem. Zo verzorgen zij faciliterende taken die niet direct te relateren zijn aan een specifieke taak die uitgevoerd moet worden. De belangrijkste groep van

systeem-level agenten zijn waarschijnlijk de planning agenten en agenten voor systeembeheer. Zij plannen en organiseren de agenten binnen een systeem en voorzien structuren voor de globale administratie van het systeem. Verder spelen zij vaak een rol in de interacties en coördinaties tussen de agenten binnen het systeem. (Papazoglou, 2001) Een tweede belangrijke groep binnen de 'systemlevel support' agenten zijn de 'security' agenten die autorisatiediensten aanbieden en instaan voor authenticatie van agenten.

In de literatuur worden in de bovenstaande alinea beschreven agenten ook wel bestempeld als 'middle agents'. Sommige auteurs behandelen het planningaspect dieper en adviseren 'matchmaker' en 'broker' agenten. De dienst van deze agenten bestaat erin andere agenten met specifieke diensten te lokaliseren. Ze zijn als het ware een tussenpersoon in agentrelaties (Decker en Mersic, 2001)

De volgende klasse van agenten bevatten agenten die instaan voor algemene activiteiten die inherent zijn aan zakelijke processen. Papazoglou (2001) definieert hier 'negotiation' en 'contracting' agenten die instaan voor de uitwisseling van gegevens om te komen tot één welbepaalde beslissing. Deze agenten bezitten protocollen die voorschrijven hoe zij dit moeten doen en hoe gereageerd moet worden in interacties met andere agenten. Nwana (1996) schetst dit in een breder perspectief en vermeldt dat het hier gaat om transactieagenten. Deze agenten zijn belast met het uitvoeren van transacties met andere agenten. Handelstransacties zijn erg talrijk en omvatten zowel informatie-uitwisseling over bijvoorbeeld tijdstippen dat vrachten moeten arriveren tot kosten voor een bepaald product. In figuur 4 is een synthese gegeven van de besproken rollen die agenten kunnen bekleden.



Figuur 4: Uitgebreide classificatie van agenten op basis van rollen

3.4 Mogelijkheden van agenten

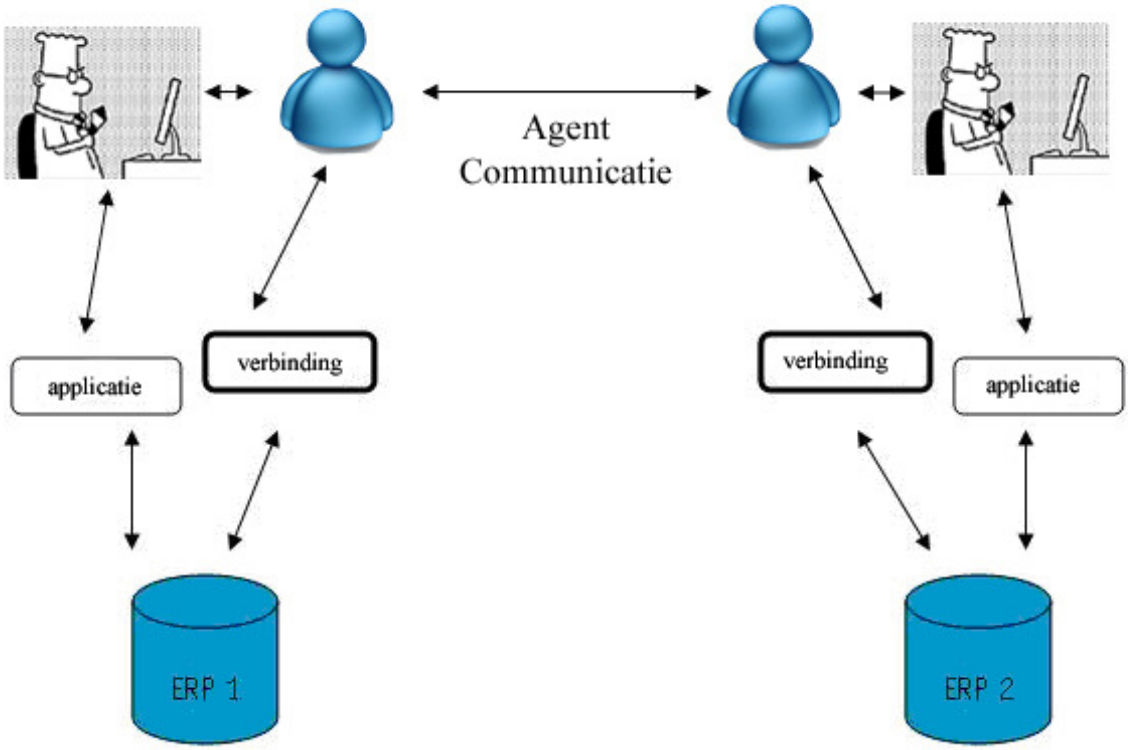
3.4.1 Agenten als zakelijke vertegenwoordigers

Gezien de kwaliteiten van agenten zijn agenten in staat om bepaalde verantwoordelijkheden van mensen op zich te nemen en deze onderling te vervullen. Zo vinden we bij Papazoglou (2001) terug dat het centrale idee en de kracht van een agent bestaat uit het feit dat de gebruiker of bezitter van een agent bepaalde taken kan delegeren naar de agent. Wortmann en Szirbik (2001) vertellen dat deze vorm van 'software component empowerment' het eerst voorkwam in commerciële aankoopagenten. Deze agenten werden ingeschakeld om bepaalde goederen te kopen in opdracht van de gebruiker. Verder vermelden zij dat de volgende stap in dit proces het ontstaan van e-market mechanismen was waar koper- en verkoperagenten onderling veilingen afhandelden. Tot slot halen zij de mogelijkheid aan om via het agentparadigma een virtueel netwerk van agenten te ontwikkelen naar het evenbeeld van de realiteit.

De rol die agenten kunnen vervullen als vertegenwoordigers wordt voorts ook bekrachtigd in andere literatuur. Zo definieert Verduijn (g.d) in dit opzicht agenten als

zakelijke vertegenwoordigers van bepaalde processen binnen een organisatie. Ook bij Gelati (2004) kunnen we lezen dat hij in de toekomst verwacht dat mensen, organisaties en processen constant online vertegenwoordigd zullen zijn door een agent en dus een soort virtuele wereld nabootsen waarbij eenvoudige taken geautomatiseerd kunnen worden. We moeten wel rekening houden met het feit dat agenten slechts kunnen omgaan met triviale processen die niet bedrijfskritisch zijn. (Wortmann en Szirbik, 2001)

Door agenten is men dus in staat om bepaalde taken waarmee menselijke interactie gemoeid is te automatiseren. Via de reeds besproken eigenschappen van agenttechnologie kunnen we inzien hoe dit mogelijk is. De autonomie die een agent karakteriseert, zorgt immers dat deze onafhankelijk van andere agenten en menselijke gebruikers kan handelen. Hij kan dus in zijn werking enkel beïnvloed worden door zijn eigenaar die hem opdrachten kan geven. Verder behartigen de agenten door hun doelgericht gedrag de belangen van de persoon of het proces dat ze vertegenwoordigen. Als vertegenwoordiger dient men te onderhandelen met externe partijen. We hebben reeds gezien dat agenten communicatievaardigheden bezitten zodat ze ook aan deze vereiste voldoen. Deze communicatie dient geïmplementeerd te worden als een asynchroon proces waardoor ‘real-time’ en parallelle verwerking mogelijk is. Een inkoopagent kan bijvoorbeeld met verschillende verkoopagenten in andere onderhandelingsfases zitten. Verder bezit een agent een bepaald zelflerend karakter en kan hij in zekere mate proactief acties ondernemen. (Verduijn, g.d.)



Figuur 5: Agenten als zakelijke vertegenwoordigers

In bovenstaande figuur is een oppervlakkige voorstelling gegeven van de manier waarop een agent een zakelijke vertegenwoordiger kan zijn. In deze concrete situatie geldt dit voor een persoon. Binnen de context van deze thesis zal een agent die de mens nauw assisteert bij zekere taken, zoals het zoeken naar informatie op het internet door een 'shopping bot' bijvoorbeeld niet relevant zijn. Vermits we onderzoeken welke voordelen agenten kunnen hebben binnen logistieke netwerken zal noodzakelijkerwijze een agent in contact treden met andere agenten zodat we het steevast over een multi-agentsysteem hebben.

3.4.2 Agenten als ondersteuning voor gedistribueerde processen

Een kenmerk van agenttechnologie dat nog niet ter sprake gekomen is, is de manier van berichtuitwisseling tussen agenten en procesafhandeling binnen een agentsysteem. Dit is te verklaren door het feit dat dit een eigenschap is van een agentsysteem in zijn geheel.

In tegenstelling tot traditionele systemen waar processen afgehandeld werden volgens een vaste volgorde die door een centrale module gestuurd werd, is de procesafhandeling in een multi-agentsysteem decentraal georganiseerd (Jennings et al., 1998; Papazoglou, 2001). Agenten zijn immers autonome en deels proactieve entiteiten en worden niet gestuurd door een centrale module.

Voorgaande impliceert dat de berichtuitwisseling tussen agenten asynchroon kan verlopen. (Anumba et al., 2001 en Fox et al., 2000 en Nissen, 2001: allen geciteerd in Verduijn et al., 2003; Jennings et al., 1998) Asynchrone communicatie doelt op het feit dat communicatie van agenten willekeurig op het initiatief van de agent zelf gebeurt en niet volgens bepaalde regels opgelegd wordt door een centrale module. Het is dus zo dat een conversatie tussen agent A en agent B zich kan afspelen terwijl deze agenten op dat moment ook conversaties voeren met andere agenten. (Verduijn et al., 2003)

Sociale mogelijkheden van agenten staan in het teken om bepaalde processen te ondersteunen. Door modularisatie en abstractie via het agentparadigma wordt één groot proces opgebroken in deelprocessen. Elke agent krijgt op die manier een nauw afgebakende verantwoordelijkheid. Deze verantwoordelijkheid om een proces te ondersteunen zal een agent opnemen door zijn doelgericht gedrag (Nwana, 1996). De complexiteit van problemen zal hierdoor dalen terwijl de snelheid, de betrouwbaarheid en de flexibiliteit zullen stijgen.

In een agentsysteem met meerdere autonome entiteiten, in dit geval de agenten, kan het zijn dat door het sociale karakter van een agent meerdere agenten betrokken worden bij een proces. Verder kan het zijn dat twee agenten op een systeem een doelgericht gedrag vertonen dat gebaseerd is op eenzelfde proces. Hoewel beide agenten volledig autonome entiteiten zijn, hebben zij dus toch een relatie omdat zij betrokken zijn bij hetzelfde proces. Op deze manier kunnen processen benaderd worden vanuit verschillende perspectieven. Een logisch gevolg van bovenstaande uiteenzetting is dat niet elke agent dient te beschikken over alle informatie binnen het systeem maar enkel deze die relevant is voor hem. Deze informatie wordt aangereikt binnen het domein waarvoor de agent verantwoordelijk is. (Jennings et al. 1998; Odill, 2005)

Het is daarom dat we kunnen stellen dat multi-agentsystemen ideaal zijn om complexe processen te ondersteunen. (Jennings et al., 1998; Odill, 2005; Wortmann en Szirbik,

2001) Complexe processen kenmerken zich door het feit dat er verschillende mogelijke oplossingen bestaan om een proces te voltooien. Deze oplossingen kunnen op hun beurt geïnitialiseerd worden door meerdere entiteiten die betrokken zijn bij het proces. Deze processen waardoor verschillende afzonderlijke partijen belast worden met de uitvoering, worden ook wel gedistribueerde processen genoemd. Vaak hebben deze entiteiten nog niet eens een gemeenschappelijk doel voor ogen maar streven deze persoonlijke objectieven na (Chaib-Draa en Dignum, 2002).

3.4.3 De schaalbaarheid en uitbreidbaarheid van agentsystemen

Multi-agentsystemen worden gekenmerkt door het afwezig zijn van een centrale controle op het systeem. De opbouw van het systeem zelf kan gebeuren via een bottom-up strategie. Wanneer de centrale component die de taken op systeemniveau verzorgt geïmplementeerd is, zou men bij een goed ontworpen systeem agenten kunnen activeren en deactiveren. Daardoor kunnen we stellen dat er weinig agenten zijn die cruciaal zijn voor het gehele agentsysteem. (Odill, 2005) Zo zal bijvoorbeeld de werking van een groot logistiek netwerk, steunend op agenttechnologie, niet verstoord worden wanneer de agent die instaat voor het afspreken van leveringsdata van een bepaalde organisatie een technisch probleem ondervindt.

Bovendien kan door agenttechnologie elk bedrijf zijn zelfstandigheid bewaren doordat het agentsysteem een soort van laag bouwt boven op het bedrijfsnetwerk. De autonome agenten van een bedrijf kunnen in staat zijn om naast interactie met menselijke gebruikers en systeemmiddelen ook met andere agenten interactie te hebben. Het feit dat de agent enkel taken zal overnemen van de gebruiker betekent overigens dat men aan de bedrijfsstructuur niets dient te veranderen.

Niet alleen de beslissingsprocedures maar ook het bewaren van data kan decentraal gebeuren. Een agent krijgt de rol van expertagent toebedeeld en is verantwoordelijk voor het beheer van een database. Op deze manier kunnen multi-agentsystemen beperkingen van gecentraliseerde systemen opheffen. Via hun mogelijkheden kunnen agentsystemen meerdere databronnen van heterogene aard samenbrengen (Jennings et al., 1998).

Bovenstaande kwaliteiten maken duidelijk dat agentgebaseerde systemen makkelijk en flexibel toepasbaar zijn omdat het zo een natuurlijke manier is om een applicatie te structureren. Verder is het mogelijk dat via een goede architectuur en functionaliteit op systeemniveau een multi-agentsysteem een makkelijk schaalbaar en uitbreidbaar geheel wordt.

Op structuur van een agentsysteem zal in een volgend hoofdstuk dieper ingegaan worden. Om uitbreidbaarheid en schaalbaarheid te bekomen dienen agenten interoperabel te zijn. Deze interoperabiliteit van een agentsysteem kan men bekomen door een standaardmanier te definiëren om berichten uit te wisselen waaraan alle agenten op het platform zich houden. Een bespreking van standaarden voor agenttechnologie zal eveneens in een volgend hoofdstuk gegeven worden.

3.5 Agent communicatie

We hebben reeds gezien dat agenten door hun sociale kwaliteiten gesofisticeerde interacties kunnen voeren. Door deze gesofisticeerde interacties onderscheiden agentsystemen zich dus duidelijk van andere softwaresystemen en manieren om een applicatie te structureren. Het is daarom dat vroeger ontwikkelde talen en protocollen voor gedistribueerde toepassingen een agentsysteem niet ten volle kunnen ondersteunen. Deze talen en protocollen zijn opgebouwd vanuit het perspectief van het proces dat zij ondersteunen. (Finin et al., 1995; Chaib-draa en Dignum, 2002) In hoofdstuk 2 beschreven we hoe we de integratie van data konden bewerkstelligen via EDI en XML. Deze integratie kon bekomen worden door uitdrukkelijke afspraken te respecteren zodat standaarddocumenten uitgewisseld konden worden. Verder zijn er applicatiestandaarden die bepalen hoe toepassingen met elkaar communiceren. Dergelijke protocollen[IV] zoals HTTP, IIOP en FTP schrijven voor hoe toepassingsspecifieke communicatie moet verlopen en rekenen via onderliggende internetstandaarden af met processen voor het transport van data en berichten. (Panko, 2005)

[IV] De term protocol wordt gebruikt om een aantal verschillende concepten te benoemen. Zo kan een protocol ook een specificatie zijn van een framework voor agentinteracties die samen een conversatie vormen. Ook wel interactieprotocol genoemd.

Het is dus duidelijk dat een communicatietaal voor agenten, een 'agent communication language' (ACL), zich op een hoger niveau dan bestaande talen bevindt. Chaib-draa en Dignum (2002) definiëren ACLs als "complex structures composed of different sublanguages that specify the message content, interpretation parameters […], the propositional attitude under which the receiver should interpret the message content, and several other components". In tegenstelling tot bovenvermelde protocollen zijn ACLs gericht op de intenties van de communicerende partijen en de sociale context waarbinnen ze verstuurd zijn. Dit wordt weer mooi vermeld in een citaat van Chaib-draa en Dignum (2002): "Typical ACLs also have a characteristic mentalistic semantics that is far more complex than standard distributed object protocols". Het is net door deze ondersteuning dat agenten in staat zijn om gesofisticeerde interacties zoals coöperatie aan te gaan.

Verder kunnen we vermelden dat agenten via deze manier van interactie de interoperabiliteit van het syntactische niveau naar semantische niveau kunnen brengen (Poggi et al., 2002). Het is immers zo dat populaire ACLs naast hun mentale semantiek ook de nadruk leggen op pragmatiek[V] (Singh, 1998). Door deze specificaties op semantisch en pragmatisch niveau kan een ACL dus algemeen verstaan worden. Wanneer een agent bijvoorbeeld een bericht stuurt met de 'communicative act' `Request` verstaat de agent die dit ontvangt dat deze agent iets van hem verwacht wat gespecificeerd is in de inhoud van het bericht[VI].

3.5.1 Opbouw van een ACL

In de definitie van een ACL zagen we dat een ACL bestaat uit verschillende subtalen. Het is immers zo dat een ACL best een gelaagde structuur bevat om zo flexibel en overzichtelijk de verschillende sublagen van ACL te kunnen gebruiken. Deze structuur bestaat uit een communicatietaal en een inhoudstaal. De communicatietaal moet een aantal primitieven of 'communicative acts' definiëren. Deze 'communicative acts' geven mening aan een bericht en zijn gebaseerd op de 'natural language speech act theory'. Dit laatste wil zeggen dat men via communiceren van deze 'communicative acts' bepaalde

[V] Pragmatiek beschouwt aspecten buiten de taal zelf en handelt in dit geval dus over de toestand van een agent en zijn omgeving waarin de interactie gebeurt. (Singh, 1998)
[VI] Hier wordt wel de assumptie gemaakt dat beide agenten de ACL delen en correct implementeren.

acties kan laten gebeuren. Deze primitieven zijn op die manier gekozen dat zij een actie tot gevolg hebben door het simpele feit dat ze verzonden worden. Dit is mogelijk doordat de ontvangende agent de mening die de verzender aan het bericht gegeven heeft kan begrijpen en er dan naar kan handelen. (Chaib-draa en Dignum, 2002) We verwijzen hier weer bij wijze van voorbeeld naar het verzenden van een `Request` bericht zoals hierboven vermeld.

In taalfilosofie worden 'communicative' of 'speech acts' onderverdeeld in een zevental soorten. De populairste ACLs voorzien vandaag slechts twee à drie van deze soorten namelijk 'representatives', (ook wel 'assertives' genoemd) en 'directives'. Via de eerste is het mogelijk een toestand mee te delen en via 'directives' is het mogelijk om aan de geadresseerde te vragen of op te dragen iets te doen. (Singh, 1998; Chaib-draa en Dignum, 2002)

De inhoudstaal wordt gebruikt om de inhoud van een bericht voor te stellen. Net zoals alle berichten dient de inhoud van een bericht een bepaalde syntax te hebben. Deze syntax wordt in een bericht opgelegd door het gebruik van een bepaalde inhoudstaal die voorschrijft hoe symbolen en tekens gerangschikt moeten worden. Buiten deze voorstelling is het ook nodig dat agenten op een zinvolle manier kunnen communiceren binnen bepaalde toepassingsdomeinen. Elk domein heeft zijn eigen woordenschat die we kunnen specificeren in een ACL bericht via een ontologie. (Finin et al., 1995; Chaib-draa en Dignum, 2002) Een ontologie is een verzameling van woorden en labels waarmee bepaalde kennis of informatie formeel kan voorgesteld worden, een soort van woordenschat dus. Bijvoorbeeld in een ontologie voor logistieke processen kan men in de woordenlijst het label "Vracht in Kilogram" opnemen. Dankzij dit concept zijn beide agenten in staat om interacties aan te gaan over een bepaald kennis- of informatie-element.

3.5.2 Beperkingen aan ACLs

Een belangrijk inzicht bij het ontwikkelen van een agentsysteem is de intieme relatie tussen enerzijds de veronderstellingen en modellering van het gedrag van agenten ook wel de 'agency' genoemd en anderzijds de semantische conventies die vastgelegd zijn in de ACL. Meer bepaald betekent dit dat de 'agency' en een ACL binnen een agentsysteem

op elkaar afgestemd moeten worden. Wanneer een agent een ACL bericht verstuurt, heeft dit door het gebruik van een 'communicative act' een vooropgestelde mening die inherent is aan de semantiek en pragmatiek van de ACL. (Singh, 1998; Chaib-Draa en Dignum, 2002; Poslad et al., 2001)

Een agent valt voor het opstellen van dergelijk ACL-bericht terug op zijn eigen interne structuur. Het kan dus zijn dat deze interne structuren niet geschikt zijn voor de ACL die gebruikt wordt. Zo halen Chaib-Draa en Dignum (2002) het voorbeeld van oprechtheid bij agenten aan. Sommige gesofisticeerde 'agencies' veronderstellen dat agenten soms valse informatie kunnen doorspelen over hun interne staat teneinde de andere partij te misleiden, zoals dit het geval is bij menselijke interacties. Toch is dergelijke mogelijkheid van een agent nutteloos als een ACL veronderstelt dat een agent altijd de juiste informatie over zijn interne status communiceert. De assumptie van oprechtheid vinden we, samen met nog veel andere vereenvoudigende assumpties, terug in de meest toonaangevende ACL's namelijk FIPA-ACL en KQML.

Gezien de grote complexiteit aan software-infrastructuur die nodig is om interactie mogelijk te maken, is een ACL slechts een manier om communicatie te verzorgen betreffende de intenties en sociale activiteiten van agenten. Er dient dus een communicatie-infrastructuur, eigen aan het agentsysteem, voorhanden te zijn die taken op systeemniveau beheert. Chaib-draa en Dignum (2002) hebben het over volgende taken: "message ordering and delivery, formatting and addressing, directory services, gateway-style translation, quality-of-service…".

3.5.3 ACL en standaardisering

Interactie tussen agenten wordt ondersteund door componenten op systeemniveau. Het is evident dat een standaardisatie van de syntax, semantiek en pragmatiek van een ACL voor meer interoperabiliteit en dus ook openheid en toegankelijkheid zal zorgen van dergelijke systemen. Via standaardisering van ACLs wordt het mogelijk dat heterogene agenten met elkaar interactie kunnen hebben en op zinvolle manier kunnen communiceren over informatie in hun omgeving of kennis die ze bezitten. Op deze manier kan een multi-agentsysteem ontwikkeld worden dat een open platform is voor

agenten die eventueel ontwikkeld zijn door verschillende ontwikkelaars. (Chaib-draa en Dignum, 2002)

Wanneer een ACL een standaard voor agenttechnologie wil zijn, is het belangrijk dat deze taal in een brede waaier van domeinen toegepast kan worden. Het is dus zo dat de semantiek van de communicatietaal, meer bepaald de primitieven, niet te specifiek mogen zijn maar eerder generalistisch. Op deze manier kunnen ontwikkelaars expressief te werk gaan in een breed gebied van domeinen. Hoewel de 'communicative acts' (primitieven) generalistisch opgevat zijn, kan men toepassingen op domeinniveau gaan ontwikkelen via de definiëring van een gemeenschappelijke ontologie. (Finin et al., 1995) Hoewel deze generalistische aanpak verkozen wordt boven een gedetailleerde voor elk domein, kunnen we stellen dat het onmogelijk is om een ACL te ontwikkelen zonder het definiëren van formele regels. (Chaib-draa en Dignum, 2002) We hebben reeds vermeld dat populaire ACLs vooral assumpties op pragmatisch vlak maken bij het gebruik van een ACL, dit is over de status van de agent en/of de omgeving waarin de berichten verstuurd worden.

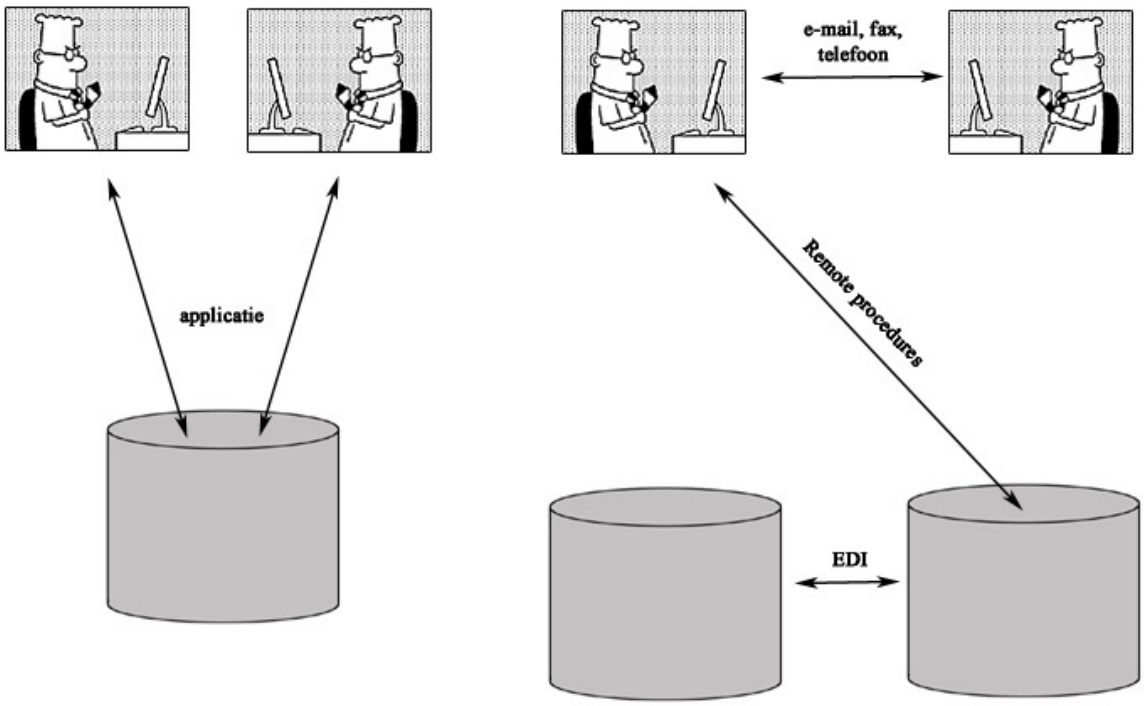
In functie van onderzoek zijn vele private agentsystemen ontwikkeld die dan gebruik maakten van communicatiemechanismen die eigen waren aan de ontwikkelaars. Aan de basis van deze mechanismen liggen semantische en pragmatische veronderstellingen die enkel gekend zijn door de ontwikkelaars van het systeem. Op die manier is het onmogelijk dat agenten van externe partijen interoperabel zijn met deze agenten vermits het onmogelijk is dat deze agenten dergelijke 'communicative acts' op uniforme manier interpreteren. (Chaib-draa en Dignum, 2002)

We kunnen dus besluiten dat een ACL slechts open en uitbreidbare systemen mogelijk kan maken wanneer duidelijke afspraken en conventies betreffende syntax, semantiek en pragmatiek gedefinieerd zijn. Verder dient domeinspecifieke interactie op elkaar afgestemd te worden via een ontologie. Standaarden op dit gebied zullen meer in detail besproken worden in de komende hoofdstukken.

3.6 Agenttechnologie als ondersteuning van logistieke netwerken

In de tekst die volgt worden conclusies getrokken uit de hoofdstukken die we reeds behandeld hebben in deze thesis, namelijk hoofdstuk 1, 2 en 3. Uit de beschrijving die terug te vinden is in de literatuur van agentsystemen en huidige toepassingen blijkt overduidelijk dat agenttechnologie beter geschikt is als ondersteuning voor processen binnen logistieke netwerken. In dit deel willen we dan ook de frappante eigenschappen en de reden blootleggen waarom agenttechnologie meer geschikt is als ondersteuning voor de logistieke sector.

Wortmann en Szirbik (2001) identificeerden twee factoren die wijzen op het grote nut van de toepassing van agenttechnologie binnen gedistribueerde en dynamische omgevingen. Ten eerste hebben zij het over de manier waarop data opgevraagd wordt door gebruikers. De nood aan data die lokaal op de bedrijfsinformatiesystemen (IS) te vinden is, is gedeeltelijk verschoven naar de nood aan informatie op andere IS buiten de onderneming. Naast de verschillende vormen van EDI gebeurt de informatie-uitwisseling tussen verschillende IS hoofdzakelijk door menselijke interacties. Bedrijven kunnen ook interfaces definiëren voor hun IS zodat externen beperkte toegang hebben tot data in dit IS. De mogelijkheden die agenten bieden in deze context zijn intuïtief duidelijk. Zo zullen agenten, zoals beschreven in de literatuur, in staat zijn om informatie uit hun omgeving uit te wisselen met andere agenten in de context van agentinteracties.



Figuur 6: Verandering in de behoefte van bedrijfsdata (eigen verwerking van Wortmann en Szirbik, 2001)

Een tweede verandering die Wortmann en Szirbik vermelden is het perspectief van bedrijfsapplicaties. Dit definiëren zij als: "the notion of scope". Deze scope duidt op de reikwijdte van bedrijfsapplicaties betreffende de processen die zij beheren. Met ERP systemen probeert men met één conceptueel datamodel elke bedrijfskundige situatie te beheren. Zoals gezien zijn deze implementaties erg inflexibel en moeizaam te voltooien. Uit dit gegeven ontsproot de nood aan mogelijkheden om marginale veranderingen door te voeren aan systemen zodat men makkelijk aanpasbare en schaalbare systemen krijgt. Via agenttechnologie echter probeert men een applicatie te structureren zodat verantwoordelijkheden en bedrijfskundige situaties beheerd worden door één enkele agent of een groep van agenten. Deze agenten worden rollen toegewezen zodat vanuit menselijk standpunt de verantwoordelijkheden op erg natuurlijke manier verdeeld worden over de agenten. Op deze manier vormen zij zelforganiserende systemen. Verder zijn de opvattingen over agentplatformen zo dat deze een makkelijke schaalbaarheid mogelijk maken.

De rollen van agenten kunnen we afstemmen op de bedrijfseconomische realiteit zodat agenten vertegenwoordigers kunnen worden van personen, processen of zelfs bedrijven.

Dit noemden we 'empowerment' van softwareapplicaties. Via de gesofisticeerde interactie die een agent kan hebben, vertegenwoordigen zij hun belanghebbenden binnen een multi-agentsysteem dat ingericht is om een bepaald doel te voltooien. Agenten benaderen dus problemen vanuit verschillende perspectieven en hebben slechts toegang tot de informatie die voor hen relevant is. Op deze manier zijn zij erg geschikt om gedistribueerde processen, zoals logistieke ketenintegratie, te ondersteunen tijdens het uitvoeren ervan. Dit is in tegenstelling tot ERP en SCM-planningssoftware die input van gegevens van het gehele proces vereisen waardoor deze software weliswaar geschikt is voor de planning van deze processen. Echter, bij het uitvoeren van deze planning dient men zich ervan bewust te zijn dat men te maken heeft met complexe relaties tussen vele belanghebbenden en de afhankelijkheid van afzonderlijke activiteiten in het proces enorm groot is. Via ERP en SCM-toepassingen wordt deze afhankelijkheid genegeerd en benadert men het proces slechts vanuit één perspectief.

Tot slot ondersteunt het agentparadigma nog enkele niet-functionele kenmerken die de flexibiliteit van multi-agentsysteem groter maken dan deze van meer traditionele systemen. Deze hebben we reeds aangehaald in dit hoofdstuk en zijn onder andere de decentrale organisatie, asynchrone afhandeling van processen en de gedistribueerde architectuur. Onderstaande tabel is een overzicht van enkele kwaliteiten die duiden op de grotere geschiktheid van agenttechnologie als ondersteuning van logistieke netwerken zoals terug te vinden in de literatuur.

Tabel 2: Vergelijking agenttechnologie met actuele toepassingen in logistieke sector

	ERP/EDI/web-platformen	**Agenttoepassing**
Procesbesturing	Centraal georganiseerd	Decentraal
Procesafhandeling	Synchroon	Asynchroon
Architectuur	Lokaal	Gedistribueerd
Vorm van interactie	Communicatie	Coöperatie
Verantwoordelijkheid	Alle processen binnen één onderneming	Elke agent is verantwoordelijk voor deelproces met strikte grenzen
Datamodel ondersteunt…	Grote complexe processen	Deelproces met strikte grenzen
Benadering proces	Eenzijdig perspectief	Gediversifieerd perspectief
Schaalbaar	Moeilijk	Makkelijk als goed platform
Mate van integratie	Data-integratie	Enterprise model integratie
Empowerment?	Nee	Ja

HOOFDSTUK 4: Standaarden binnen agenttechnologie

We hebben reeds in voorgaand hoofdstuk kort een aanzet gegeven tot het belang van standaarden voor agenttechnologie. In dit hoofdstuk gaan we dieper in op standaardisering en de rol voor agenttechnologie.

4.1 Waarom standaarden?

Het belang van een standaardisering van agentcommunicatie kunnen we afleiden aan de hand van volgend citaat van Bellifemine (2003): "When an agent is the object of a communicative action (i.e. when it receives a message), it must be able to properly understand the meaning of that action and, in particular, why that action has been performed (i.e. the communicative intention of the sender of the message)." Het is dus belangrijk dat bepaalde afspraken gemaakt worden betreffende de syntax en de semantiek van berichten die agenten uitwisselen.

Naast deze verhoging van interoperabiliteit zijn er nog andere voordelen van standaarden binnen agenttechnologie. Het is immers door deze standaardisering dat de nadruk van het onderzoek naar manieren om agentsystemen te ontwikkelen verschuift naar het ontwikkelen zelf van agentsystemen. Dit omdat ontwikkelaars door de aanwezige standaarden concrete richtlijnen hebben die ze kunnen volgen. Verder streven ontwikkelaars naar een zo groot mogelijke universaliteit van hun protocollen en ACLs. Op deze manier hopen ze een kritische massa aan gebruikers te bereiken zodat hun standaard levensvatbaar kan blijven. Het is immers de gebruiker die streeft naar eenvoud en universaliteit en daardoor de populariteit van een standaard bepaald. (Poslad et al., 2001)

4.2 Voor welke aspecten kan men standaarden ontwikkelen?

Standaardisering van agenttechnologie beslaat niet alle aspecten die eigen zijn aan de interactie tussen twee communicerende agenten. Een applicatie die volgens het agentparadigma gestructureerd is, zal immers gebruik gaan maken van reeds bestaande infrastructuren of protocollen, bijvoorbeeld voor het transport van berichten. De standaardisering van agenttechnologie zelf beperkt zich dus tot: "the interpretation and handling of ACL messages, facilitator agents, and the use of, but not the actual specification of an existing software infrastructure such as message transport protocols, and message persistence schema, to underpin agent communication" (Poslad et al., 2001). We zien dus dat binnen agenttechnologie niet enkel de materie betreffende de interpretatie, afhandeling en uitwisseling van berichten gestandaardiseerd kan worden. Er is immers ook de integratie van software-infrastructuren en de interactie met bepaalde componenten op systeemniveau binnen een agentsysteem die volgens bepaalde procedures dienen te verlopen.

Zoals reeds gezegd kunnen systeemcomponenten als centrale of als decentrale diensten geïmplementeerd worden. Hun functionaliteit is vooral het ondersteunen van faciliterende taken zoals onder andere de ondersteuning van berichtuitwisseling en het beheren van een centrale index voor diensten en agenten. Hierbij vormen deze componenten vaak een brug tussen de gedistribueerde agenten onderling maar ook tussen het agentsysteem en andere software-infrastructuren zoals transportprotocollen.

De interpretatie en afhandeling van een bericht dient op drie niveaus op elkaar afgestemd te worden. De eerste laag is de laag die de syntax omhelst. Symbolen dienen in een gestructureerde manier geplaatst te worden net zoals dat het geval is in menselijke talen. Een andere laag die van belang is in berichtuitwisseling is de overeenstemming van applicaties over de semantiek van bepaalde constructies. Hetzelfde concept, object of entiteit moet een uniforme mening krijgen voor verschillende applicaties. Deze materie wordt behandeld in een ontologie[VII]. De derde laag gaat over het modelleren van voorstellen, gedragingen en attitudes in een bericht. Deze vereiste wordt ondersteund door de mentale semantiek die aan de grondslag ligt van een ACL. We zien dus een opbouw van de uit te wisselen berichten in drie lagen. Deze lagen

[VII] Een ontologie is een een taxonomie van termen met hun definities en grondslagen is. We kunnen het vergelijken met een woordenboek.

kunnen op functioneel gebied onafhankelijk van elkaar geïmplementeerd worden. (Labrou et al., 1999)

De twee belangrijkste specificaties voor agent standaardisatie zijn KQML en FIPA. In de volgende delen zullen we deze standaarden in detail bespreken.

4.3 Advanced Research Projects of Agency (ARPA)

KQML of de "Knowledge Query and Manipulation Language" was het resultaat van een eerste poging om te komen tot een gestandaardiseerde ACL en is ontwikkeld door het ARPA Knowledge Sharing Effort (KSE) project. Naast de communicatietaal KQML specificeerde Finin et al. (1995) ook een taal om de inhoud van berichten voor te stellen, ook wel inhoudstaal genoemd. Deze inhoudstaal doopte men de Knowledge Interchange Format (KIF) en is de oplossing die door KSE voorgesteld wordt om de syntactische aspecten binnen een communicatietaal op te vangen. In een volgende paragraaf gaan we verder in op de werking van KQML. Een verregaande gedetailleerde uitwerking van KIF zou ons echter te ver leiden in deze eindverhandeling. Meer informatie over KIF is dan ook te vinden in Finin et al. (1995) en de site van het UMBC (UMBC AgentWeb, 2007a).

4.3.1 KQML

Finin et al. (1995) omschrijven KQML in drie lagen: "the content layer, the message layer, and the communication layer". De 'contentlayer' is eerder een technisch aspect in die zin dat het de echte inhoud van het bericht bevat in een bepaald formaat zoals ASCII strings of binaire strings. Het hart van KQML is de berichtlaag waar de 'communicative act'[VIII] gedefinieerd wordt. Deze 'communicative act' wordt vergezeld van belangrijke parameters om de inhoud van het bericht te begrijpen zoals de gebruikte ontologie en de inhoudstaal. In de communicatielaag tot slot vinden we parameters terug die de identiteit van de zender, ontvanger en een identificatienummer voor het bericht zelf weergeven.

Onderstaand voorbeeldje van een bericht uit Finin et al. (1995) kan waarschijnlijk voor een beter begrip zorgen. In dit bericht is de 'performative' van het bericht `ask-one` en

[VIII] Een 'communicative act' wordt door Finin et al. (1995) ook wel een 'performative' genoemd.

worden parameters ingegeven na een dubbelpunt. Zo kunnen we afleiden dat de `:content` opgemaakt is in de `:language` 'LPROGLOG' en de `:ontology` 'NYSE-TICKS' gebruikt. Het bericht dat hier verstuurd wordt door agent 'joe' is een vraag aan de 'stock-server' hoeveel de prijs van een aandeel van IBM is. De ontologie is dan ook waarschijnlijk gedefinieerd op het domein van beurzen, in dit geval de NYSE.

```
(ask-one
      :sender joe
      :content (PRICE IBM ?price)
      :receiver stock-server
      :reply-with ibm-stock
      :language LPROLOG
      :ontology NYSE-TICKS)
```

De mogelijke 'performatives' van de KQML zijn gedefinieerd in onderstaande tabel. De kolom 'Category' leert ons het algemeen nut van het versturen van de 'performatives' die rechts opgesomd worden. Wat wel belangrijk is en beklemtoond wordt, is dat deze verzameling uitbreidbaar is. Er kunnen dus naar believen 'performatives' toegevoegd worden. De interoperabiliteit blijft echter slechts gegarandeerd als agenten die met elkaar communiceren overeenstemming vinden betreffende de interpretatie van deze nieuwe 'performatives'.

Tabel 3: Performatives van KQML (Finin et al., 1995)

Category	*Name*
Basic query	evaluate, ask-if, ask-about, ask-one, ask-all
Multi-response (query)	stream-about, stream-all, eos
Response	reply, sorry
Generic informational	tell, achieve, cancel, untell, unachieve
Generator	standby, ready, next, rest, discard, generator
Capability-definition	advertise, subscribe, monitor, import, export
Networking	register, unregister, forward, broadcast, route,

4.3.2 Kritieken op KQML

Ten eerste is KQML geen standaard in de breedste zin van het woord in dat opzicht dat er in de agentwereld geen enkele formele consensus bestaat over bepaalde specificaties. (Poslad et al., 2001) KQML is immers het resultaat van de inspanningen van een select

groepje van onderzoekers. Verder kunnen we opmerken dat het oorspronkelijk bedoeld was als interface voor informatieoverdracht tussen kennisdatabanken. Oorspronkelijk hadden de KQML specificaties dus weinig met agenttechnologie te maken.

We hebben reeds besproken in hoofdstuk 3 dat er een nauwe band is tussen de 'theory of agency' en het gebruik van een ACL. Welnu, KQML is ontworpen met in het achterhoofd een 'agency' die gebaseerd is op het feit dat agenten beheerders zijn van hun eigen virtuele database[IX]. Berichten versturen of ontvangen viel dus samen met een opvraging of versturing van de gegevens uit deze database. Deze 'agency' is relatief eenvoudig en veronderstelt basisalgoritmes en taken voor een agent. Het is ook daarom dat de semantiek van KQML-berichten niet formeel gedeclareerd werd en het grootste deel van de semantische waarden die in deze berichten vervat zitten dus via onderlinge afspraken tussen ontwikkelaars bepaald moesten worden. (Chaib-draa en Dignum, 2002) Dit fenomeen had natuurlijk repercussies voor de interoperabiliteit daar afzonderlijke agentsystemen andere basisveronderstellingen betreffende de semantische waarden van berichten er op nahielden. Ook Singh (1998) bestempelt dit gebrek aan formele semantiek als negatief voor de graad van interoperabiliteit. Meer nog, hij vermeldt dat dit het ontstaan van zogenaamde dialecten in de hand werkt wat wordt gestaafd aan de hand van vele papers die gewijd zijn aan een of ander dialect van KQML. (Chaib-draa en Dignum, 2002) Voorbeelden hiervan vinden we in Dragoni et al. (2006) en Sultan en Nien (2006).

Hoewel in de specificaties van Finin et al. (1995) ook aandacht wordt geschonken aan de interactie van agenten met faciliterende diensten op systeemniveau, wordt hier geen uitgesproken standaard voor gedefinieerd. Het is eerder zo dat de groep achter de KQML specificaties slechts suggesties doet om deze interacties te laten verlopen. We kunnen dus stellen dat het KQML ook hier weer ontbreekt aan formele richtlijnen.

Tijdens de eerste jaren na de publicatie van de KQML-standaard in Finin et al. (1995) beseften de auteurs dat het gebrek aan formele semantiek het gebruik van hun standaard niet ten goede komt. Het is daarom dat ze formele semantiek toegevoegd hebben aan KQML. (Labrou, 1996; Labrou en Finin, 1997; Labrou en Finin, 1998) Met deze actie werd de werkwijze van die andere organisatie die zich inzet voor

[IX] Deze agency was immers het meest plausibel daar het oorspronkelijke opzet van KQML het informatie uitwisselen tussen verschillende databases was.

interoperabiliteit binnen de agentwereld, namelijk FIPA, benaderd. We zullen deze organisatie en zijn werkwijze dan ook bespreken verderop in dit werk.

De semantische definities die werden toegevoegd aan elke KQML-performatives formuleerden 'preconditions', 'postconditions' en 'completionconditions'. De 'preconditions' van een 'peformative' definiëren de noodzakelijke toestand van een agent om een 'performative' te ontvangen of te versturen. De 'postconditions' beschrijven de toestand van de verzendende agent nadat deze het bericht verstuurd heeft. De 'completion conditions' tot slot is een indicatie voor de uiteindelijke status van de agenten na een succesvolle berichtuitwisseling. Een succesvolle berichtuitwisseling betekent dat de intentie die er bij een bepaalde agent was om de berichtuitwisseling op te starten, bevredigd is.

Om dit concreter voor te stellen kunnen we de vergelijking maken met menselijke communicatie. Een persoon die erg blij is, zal geen verdrietige taal spreken omdat dit niet eigen is aan de gemoedstoestand waar hij zich in bevindt. De vereiste 'precondition' voor een verdrietig bericht is dus dat deze persoon zich ook verdrietig voelt. De 'postcondition' bij het verzenden van een verdrietig bericht kan zijn dat een persoon zich opgelucht voelt omdat hij zijn verdriet heeft kunnen uiten. De 'completion condition' die gedefinieerd zou kunnen worden wanneer een andere persoon antwoordt op een verdrietig bericht is dat de zender zich getroost voelt.

We kunnen nu ook verklaren hoe het komt dat de 'agency' en het gebruik van een ACL zo nauw bij elkaar aansluiten. Om aan de verschillende condities te kunnen voldoen, dient er immers in de interne structuur van de agent de mogelijkheid te bestaan om deze toestand vast te leggen. Labrou et al. (1999) zeggen hierover het volgende: "Preconditions, postconditions, and completion conditions describe states of agents in a language of mental attitudes (belief, knowledge, desire, and intention) and action descriptors (for sending and processing a message)". We kunnen hieruit besluiten dat in de interne structuur van een agent mentale attitudes zoals 'beliefs', 'knowledges', 'desires' en 'intentions' vervat moeten zitten. Het is evident dat deze interne structuur zal overeenkomen met de BDI-structuur die we reeds aangehaald hebben in deze thesis in hoofdstuk 3.

4.4 FIPA

We zullen in eerste instantie de context van de FIPA-standaarden schetsen door de werkwijze van de organisatie, een vergelijking met KQML en de kenmerken van deze standaarden te schetsen. Tot slot geven we dan in een volgend onderdeel de concrete bespreking van de FIPA-standaarden.

De Foundation for Intelligent Physical Agents (FIPA) is een open non-profit organisatie die als doel het promoten van agenttechnologie heeft. De leden van de organisatie zijn zowel academische onderzoeksgroepen als grote industriële spelers zoals Toshiba Corp. en Siemens. Verder ontwerpt FIPA specificaties over agenttechnologie die interoperabiliteit tussen agentsystemen proberen te verhogen. Deze specificaties worden ontwikkeld door verschillende comités die verantwoordelijk zijn voor de specificaties van de ACL, agentdiensten op het platform en integratie van agenten met andere software-infrastructuren. (FIPA Website, 2007a; Labrou et al., 1999)

De FIPA specificaties worden op een iteratieve manier ontwikkeld door nauwe betrokkenheid van de leden in de organisatie. Zo kunnen specificaties verschillende statussen van uitvoeringen hebben zoals 'Preliminary', 'Experimental', 'Standard', 'Deprecated' of 'Obsolete'. Dit kan gezien worden als een levenscyclus van de specificaties. Eerst wordt een specificatie toegepast en aangepast zodat het kan evolueren naar een standaard. Komt men later tot een verbeterd inzicht of is de specificatie verouderd gaat de standaard 'afgeschreven' worden en uiteindelijk 'overbodig' worden. (FIPA Website, 2007a) Op de FIPA webstek[X] heeft u toegang tot alle reeds ontwikkelde standaarden. Een overzicht van de standaard specificaties vindt u in bijlage II.

4.4.1 FIPA in vergelijking met KQML

De manier van werken van FIPA als een organisatie en de manier om te komen tot standaarden is anders dan bij KQML. We zien dat FIPA een organisatie is die in tegenstelling tot de werkgroep rond KQML wel via consensus tot standaarden kan komen.

[X] http://www.fipa.org/specs/index.html

Verder is het zo dat het domein van de FIPA-specificaties veel verder reikt dan slechts een ACL. De FIPA standaarden zijn te verdelen over twee types namelijk de individuele componenten en de interacties en verbanden tussen deze componenten. (Poslad et al., 2001) Op deze manier probeert FIPA haar abstracte specificatie van het agentplatform (AP), uitgelegd in onderdeel 6.1 van dit hoofdstuk, uit te werken.

De FIPA-ACL wordt door FIPA ook via een gelaagde structuur opgebouwd zoals dit het geval was bij KQML. Verder kunnen we vermelden dat FIPA altijd strenge semantische regels gedefinieerd heeft voor het gebruik van zijn ACL. Vele critici verkiezen deze werkwijze boven het eerder losse karakter van de semantische regels van KQML. (Singh, 1998)

De semantische regels van de FIPA-ACL beschrijven de 'feasibility preconditions' en het 'rational effect' van de 'communicative acts'. Als een agent een bepaalde 'communicative act' wilt uitvoeren door een bericht te sturen dient hij te voldoen aan de 'feasibility preconditions'. Het 'rational effect' van een 'communicative act' is de vooropgestelde invloed dat het verstuurde bericht gaat hebben op de ontvangende agent. Deze 'feasibility preconditons' en het 'rational effect' worden uitgedrukt in 'beliefs', 'desires', 'uncertain beliefs', en 'intentions'. (Labrou, 1999; Chaib-draa en Dignum, 2002) De informatie bij de bespreking van de semantische concepten betreffende KQML kunnen ook aangewend worden om een beter begrip te krijgen van de semantische definities betreffende de FIPA-ACL daar de opbouw van beide talen erg gelijkend is. (Labrou et al., 1999)

4.4.2 Waarom de bespreking van de FIPA standaarden?

Het is de globale aanpak van FIPA die het zo interessant maakt om de standaarden verderop in dit hoofdstuk toe te lichten. Naast het feit dat deze standaarden een antwoord zijn op de algemene nood aan standaardisatie is een uitgebreide bespreking van de FIPA standaarden nog om andere redenen nuttig. De FIPA standaarden kunnen immers een meerwaarde betekenen omdat ze de lezer een idee verschaffen hoe de beschreven eigenschappen van agenten en multi-agentsystemen, die we reeds op een abstract niveau in hoofdstuk 3 beschreven hebben, meer concreet ingevuld kunnen

worden. In die zin kunnen de FIPA standaarden dienen als een referentiemodel voor de bouw van een agentapplicatie.

Verder is agenttechnologie een technologie die gekenmerkt wordt door een lage maturiteit. Binnen dit domein zijn er dus nog geen heiligmakende methodologieën, tools of complementaire diensten en toepassingen beschikbaar (Luck et al., 2005). Binnen dit geheel is JADE, dat aan de FIPA-specificaties voldoet, één van de meest succesvolle agentplatformen. Omdat we later JADE gaan bespreken is een basisbegrip van FIPA vereist.

4.5 Kritiek op de BDI-agency focus

We hebben het reeds kort aangehaald: de semantische inhoud van een ACL is gerelateerd aan de opvattingen die een ontwerper heeft over het gedrag van de agent. De semantische basis van FIPA-ACL gaat, net als die van KQML, uit van de opvatting dat een agent 'beliefs', 'desires' en 'intentions' heeft. In de literatuur wordt hiernaar verwezen als het BDI- of mental agency. (Chaib-Draa en Dignum, 2002) Op deze opvatting waarop de bekendste ACLs namelijk FIPA-ACL en KQML gebouwd zijn, zijn toch wel enkele kritieken terug te vinden in de literatuur.

De mening die aan een FIPA-ACL bericht gegeven wordt, is vanuit een privaat perspectief van een agent bekeken. Wanneer een agent een bericht verstuurt of ontvangt weet hij dus helemaal niet hoe dit bericht invloed heeft op de mentale toestand, uitgedrukt in BDI, van de andere partij. De mening van een 'communicatieve act' is dus gebaseerd op de intentie van de ontvanger of de interpretatie van de zender. In praktijk blijkt overigens dat het programmeren van een mentale toestand vaak niet eenduidig kan gebeuren. Communicatieve acts gaan dan fout gebruikt worden. Zo is er de tendens bij KQML om alle communicatieve acts te definiëren als varianten op de 'performative' `tell`. Op die manier kunnen we stellen dat het niet veel uitmaakt welke 'performative' gebruikt wordt. Doordat de mening van een bericht op een private manier bepaald wordt door elke agent, verliezen de 'performatives' dus veel van hun waarde. (Singh 1998)

Bovendien worden over de BDI-status assumpties gemaakt door de ontwikkelaars van de ACL. Frequent aangehaald in de literatuur is de assumptie van de oprechtheid die KQML

en FIPA-ACL hanteren. Om agenttechnologie te ondersteunen is dit eerder nadelig, er wordt immers een deel van de autonomie weggenomen. (Singh 1998) Ook in de tekst van Chaib-draa en Dignum (2002) vinden we commentaren over deze assumptie terug. Zij voegen er nog aan toe dat een ACL zo weinig mogelijk op assumpties gefundeerd moet worden teneinde de interoperabiliteit te vrijwaren. Toch is het volgens hen onmogelijk om een ACL zonder assumpties te ontwikkelen.

Vele auteurs ijveren dan ook voor een alternatief voor de 'BDI-agency' als basis voor een ACL, namelijk het 'social agency' (Chaib-Draa en Dignum, 2002; Singh, 1998). Deze benadering gaat ervan uit dat elk bericht gesitueerd moet worden in de context van een conversatie. Berichten worden dan uitgewisseld op basis van voorgeschreven uitwisselingsprotocollen waarbij agenten rollen worden toebedeeld. Via deze opvatting worden verzonden en ontvangen 'communicatieve acts' ook gekaderd binnen een breder kader van uitwisselingen zodat de conventie, dit is de gezamenlijke overeenstemming, van een bepaald bericht gekend is door zender en ontvanger. (Singh, 1998) We zien dus dat FIPA door het ontwikkelen van de interactieprotocollen het gebruik van ACLs probeert te kaderen binnen een context. Op deze manier vangen zij tekortkomingen op die traditionele ACLs kenmerken en heeft de FIPA-ACL een streepje voor op KQML omdat zij ondersteund wordt door deze interactieprotocollen.

4.6 FIPA standaarden

In het resterende gedeelte van dit hoofdstuk wordt de samenhang tussen de componenten en standaarden die reeds ontwikkeld zijn door FIPA geschetst. Op deze manier hoop ik dat de lezer snel op weg gezet kan worden om de FIPA standaarden te begrijpen en de documenten makkelijk te kunnen lezen. Tot slot kijken we al een beetje in de toekomst naar veelbelovende ontwikkelingen in de FIPA standaarden.

Deze bespreking is integraal opgebouwd aan de hand van de documenten die online raadpleegbaar zijn op de FIPA website. Elk specificatie is door FIPA geïdentificeerd met een identificatienummer dat we dan ook zullen vermelden in deze tekst tussen vierkante haken. Via dit uniek identificatienummer is het immers erg makkelijk een bepaalde specificatie terug te vinden in de verzameling van FIPA-specificaties.

4.6.1 FIPA Abstract Architecture [00001]

Zoals reeds gezegd streeft FIPA interoperabiliteit en herbruikbaarheid na. Dit kan enkel door in eerste instantie de elementen waaruit een toekomstige architectuur zou moeten bestaan te identificeren. Deze elementen moet men verder op elkaar afstemmen via het specificeren van een ACL, een ontologie en eventueel interactieprotocollen. Deze standaarden kunnen in een volgend stadium, namelijk het ontwerp, geconcretiseerd worden tot bepaalde objecten en functies van een agentsysteem.

FIPA definieert een drietal abstracte diensten die aanwezig moeten zijn op een AP. Deze drie diensten worden gedefinieerd als `agent-directory-service`, `message-transport-service` en `service-directory-service`.

De eerste dienst namelijk de `agent-directory-service` heeft tot doel om agenten te identificeren en lokaliseren. In die zin kunnen we stellen dat het de 'white-page'-dienst van het AP omhelst. Via deze dienst kunnen agenten zichzelf en hun locatie registeren en andere agenten en hun locatie opzoeken.

De `message-transport-service` staat in voor het verzorgen van agentcommunicatie. FIPA identificeert drie cruciale aspecten van communicatie zijnde de berichtstructuur, berichtvoorstelling of codering en het berichttransport. Een bericht is geschreven in een ACL. De inhoud van dergelijk bericht wordt voorgesteld in een 'content-language' of inhoudstaal. In de content maakt men gebruikt van ontologieën die in feite de woordenschat voor een specifiek domein afspreken. Betreffende het transport van een bericht moet men een onderscheid maken tussen de te verzenden inhoud en de informatie die nodig is voor het transport zelf.

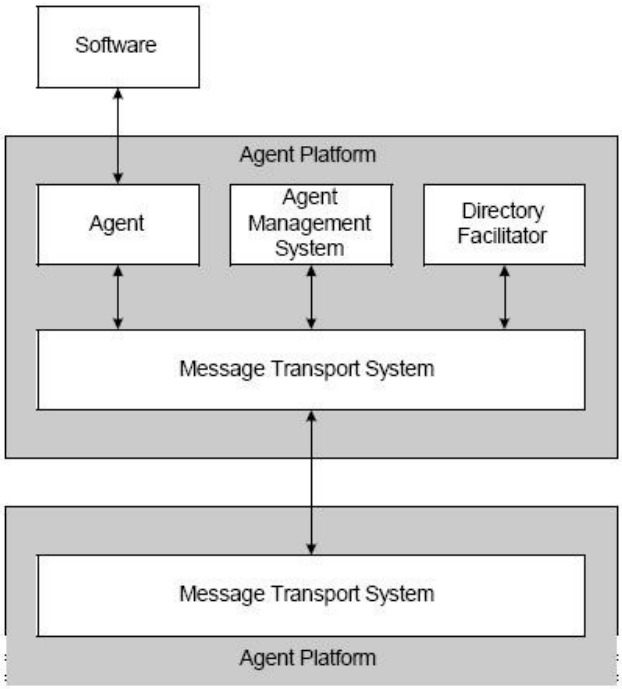
Tot slot is er de `service-directory-service` die de mogelijkheid biedt aan agenten om op een consequente manier diensten te zoeken en te ontdekken. Deze dienst voorziet een centrale component waar diensten geregistreerd en gezocht kunnen worden. Deze dienst wordt ook wel de 'yellow-pages'-service genoemd.

Een belangrijk onderscheid dat gemaakt moet worden, is het verschil tussen de agentcommunicatie zelf en het transport van agentberichten. Zoals we reeds gezien hebben kunnen agenten op een semantisch niveau communiceren via ACLs. Verder

dienen agentberichten gebruik te maken van een middel om het fysisch transport van een bericht te verzorgen. Onder deze middelen bevinden zich standaarden zoals HTTP, IIOP en Java RMI.

4.6.2 FIPA Agent Management Specificatie [00023]

FIPA heeft door zijn ervaring met agenten een agent-framework opgesteld. Dit agent-framework is zo opgebouwd dat agenten er op een effectieve en efficiënte manier kunnen opereren. FIPA definieert diensten die ingericht zijn op een AP en beschikbaar zijn voor de agenten. Het is dus zo dat er andere manieren zijn om een agentsysteem te structureren maar aangezien FIPA als referentiemodel gekozen werd, houden we ons aan deze structuur. In onderstaand voorbeeld word dit standaardmodel van diensten op een agentsysteem geschetst.



Figuur 7: FIPA agent-framework (FIPA)

Een agentplatform is volgens FIPA de fysische infrastructuur waarop agenten actief kunnen zijn. Een agentplatform zijn dus in feite de machines, besturingssystemen, ondersteunende software, agentmanagement componenten (dit zijn de Directory Facilitator, het Agent Management System en het Message Transport System) en de

agenten zelf. Het ontwikkelen van de specificaties voor een volledig AP valt buiten de bevoegdheid van een standaardisatiegroep en behoort eerder tot de verantwoordelijkheid van de programmeur en administrator van de toepassing.

Een Directory Facilitator (DF) is een dienst die aangeboden wordt aan agenten die op het systeem verblijven. Via deze dienst kunnen agenten diensten aanbieden aan andere agenten of diensten opzoeken. Op deze manier concretiseert het de `service-directory-service`, die we kennen van de abstracte architectuur van FIPA en die de 'yellow-pages' van een AP zijn. De acties die een DF kan uitvoeren zijn 'register', 'deregister', 'modify' en 'search'. Om interactie aan te gaan met een DF dient een agent het `fipa-request` interactieprotocol te gebruiken [FIPA 00026]. Verder kan een agent zich inschrijven zodat hij op de hoogte gehouden blijft van eventuele registraties, deregistraties of aanpassingen voor bepaalde diensten die geregistreerd zijn in de 'yellow-pages'. Om deze inschrijving mogelijk te maken moet een DF het `fipa-subscribe` interactieprotocol [FIPA00035] implementeren.

Een Agent Management System (AMS) is een verplichte component van een AP. De AMS is als een soort administrator voor het AP. Dit houdt in dat er altijd één en slechts één AMS is per AP. De belangrijkste taak voor een AMS is een directory van de Agent IDentifiers (AIDs) bijhouden. Op deze manier biedt de AMS een 'white-page' dienst aan voor andere agenten en implementeert het de abstracte concepten van de `agent-directory-service`.

Via de Message Transport System (MTS) module kunnen agenten communiceren met elkaar. We behandelen deze dienst uitgebreid verderop in dit hoofdstuk (4.6.4) nadat we in detail de FIPA-ACL besproken hebben.

Verder vinden we in de specificatie voor agentmanagement de `fipa-agent-management` ontologie terug. In deze ontologie worden de parameters vastgelegd om objecten, functies en 'exceptions' die deel uitmaken van een agentplatform te beschrijven. Een voorbeeld van een object waarvan in onderstaande figuur de parameters staan, is een 'agent-identifier' (AID). Deze AID is een unieke waarde die elke agent verschillend maakt.

Frame Ontology	agent-identifier fipa-agent-management			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
name	The symbolic name of the agent.	Mandatory	word	df@*hap_name* ams@*hap_name*
addresses	A sequence of ordered transport addresses where the agent can be contacted. The order implies a preference relation of the agent to receive messages over that address.	Optional	Sequence of url	
resolvers	A sequence of ordered AIDs where name resolution services for the agent can be contacted. The order in the sequence implies a preference in the list of resolvers.	Optional	Sequence of agent-identifier	

Figuur 8: Definiëring AID in de fipa-agent-management ontologie (FIPA)

Een AID wordt gespecificeerd aan de hand van drie parameters, namelijk `name`, `adresses` en `resolvers`. De parameter `name` geeft een unieke naam aan een agent. Via de `adresses` parameter kan een agent een link maken met het Message Transport System van het agentplatform. De `resolvers` parameter bevat extra transportadressen waarlangs berichten naar de agent gestuurd kunnen worden. In een ACL-bericht wordt een AID als volgt voorgesteld:

```
(agent-identifier
     :name agent-b@bar.com
     :resolvers (sequence
           (agent-identifier
                 :name ams@foo.com
     :addresses (sequence iiop://foo.com/acc))))
```

4.6.3 FIPA ACL Messages

Een FIPA ACL-bericht bevat een of meerdere parameters naargelang de situatie. De voorgestelde FIPA-parameters kunnen aangevuld worden met parameters die programmeurs zelf kunnen definiëren. Door deze aanvulling voldoen de berichten echter niet meer aan de FIPA vereisten zoals gedefinieerd in de FIPA ACL Message Structure [00061]. In deze specificatie wordt een overzicht gegeven van de verschillende parameters, ook wel slots genoemd, in een FIPA-ACL bericht. Een overzicht vindt u terug in onderstaande tabel 4.

Tabel 4: Parameters van een FIPA-ACL bericht (FIPA)

Parameter	Category of Parameters
performative	Type of communicative acts
sender	Participant in communication
receiver	Participant in communication
reply-to	Participant in communication
content	Content of message
language	Description of Content
encoding	Description of Content
ontology	Description of Content
protocol	Control of conversation
conversation-id	Control of conversation
reply-with	Control of conversation
in-reply-to	Control of conversation
reply-by	Control of conversation

Verder ontwikkelde FIPA een standaard die voorschrijft hoe de structuur van de 'Communicative Act Library' eruit moet zien. Deze standaard definieert de labels die samenhangen met de activiteiten die agenten uitvoeren. Deze labels vormen, net als de 'performatives' bij KQML, de basis van de ACL omdat de 'communicative acts' gelinkt worden aan een semantische betekenis die door alle agenten uniform geïnterpreteerd worden. De 'performatives' die in FIPA Communicative Act Library Specification [00037] beschreven worden, worden weergegeven in onderstaande tabel:

Tabel 5: 'Communicative acts' in de FIPA-ACL

• Accept Proposal	• Propagate
• Agree	• Propose
• Cancel	• Proxy
• Call for Proposal	• Query If
• Confirm	• Query Ref
• Disconfirm	• Refuse
• Failure	• Reject Proposal
• Inform	• Request
• Inform If	• Request When
• Inform Ref	• Request Whenever
• Not Understood	• Subscribe

De eigenlijke inhoud (content) van een 'communicative act' verwijst naar een boodschap die een bepaalde agent wil overbrengen. Via de FIPA SL Content Language Specifiation [00008] worden grammaticale en lexicale regels voorgeschreven om de inhoud op te bouwen. De naam van deze content language is `fipa-sl` en wordt dus in de `language` parameter van een ACL-bericht weergegeven.

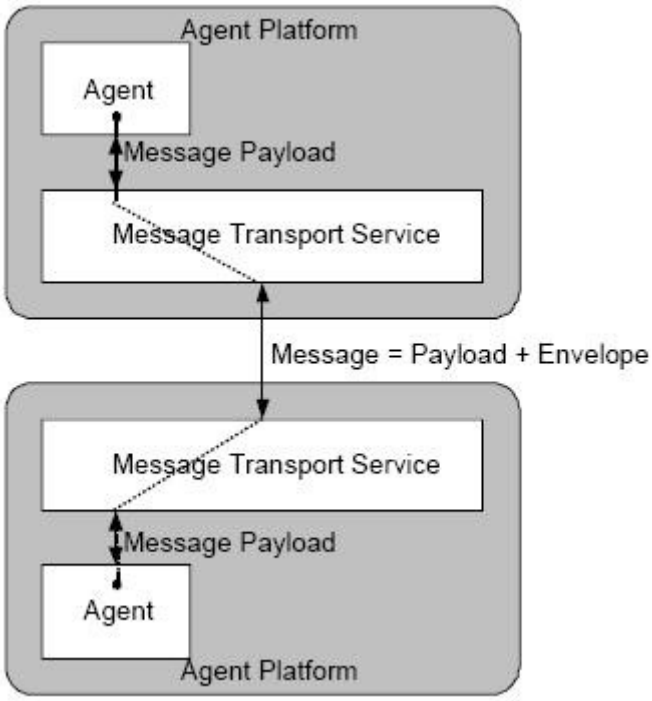
Met bovenstaande standaarden, zijnde de voorgeschreven structuur, de gebruikelijke 'communicative acts' , een inhoudstaal en een ontologie[XI] is het mogelijk een ACL-bericht op te bouwen. Deze ACL-berichten dienen in een formaat te staan zodat het getransporteerd kan worden door de Message Service van een AP. Gangbare formaten waarin ACL-berichten kunnen opgemaakt worden zijn XML en string. De gecodeerde versie van het ACL-bericht dat klaar is om verstuurd te worden noemt FIPA de 'payload'. Hoe een ACL-bericht in XML of in string-syntax kan opgemaakt worden is gedocumenteerd in de standaarden [00071] en [00070].

We merken tot slot nog de gelaagde structuur van de FIPA-ACL op, die identiek is aan deze van KQML. De FIPA ACL Message Structure [00061] definieert immers de opbouw van een bericht en bepaalt de parameters die ingevuld dienen te worden zoals de gebruikte ontologie en inhoudstaal. Verder wordt er in de FIPA Communicative Act Library Specification [00037] bepaald welke 'performatives' kunnen gebruikt worden om mening aan een bericht te geven. Tot slot hebben we in de voorgaande alinea duidelijk gemaakt hoe FIPA de codering van berichten voorschrijft.

4.6.4 FIPA Message Transport Service [00067]

Deze specificatie wil een voorbeeld zijn voor hoe de agent '`message transport service`' (MTS), voorzien wordt door FIPA. We hebben reeds gewezen op het feit dat het MTS een onderdeel is van de FIPA specificatie voor agentmanagement (zie 4.6.2). Deze dienst op niveau van het AP verzorgt het sturen en ontvangen van berichten tussen agenten. Enerzijds is in deze specificatie beschreven hoe berichten uitgewisseld kunnen worden op één agentsysteem. Anderzijds vinden we hier ook terug hoe berichten uitgewisseld kunnen worden tussen een agent binnen een bepaald systeem en een entiteit (al dan niet agent) buiten dit systeem.

[XI] Deze blijft binnen de FIPA specificaties wel beperkt tot een ontologie op het domein van agentmanagement.



Figuur 9: Message transport service (FIPA)

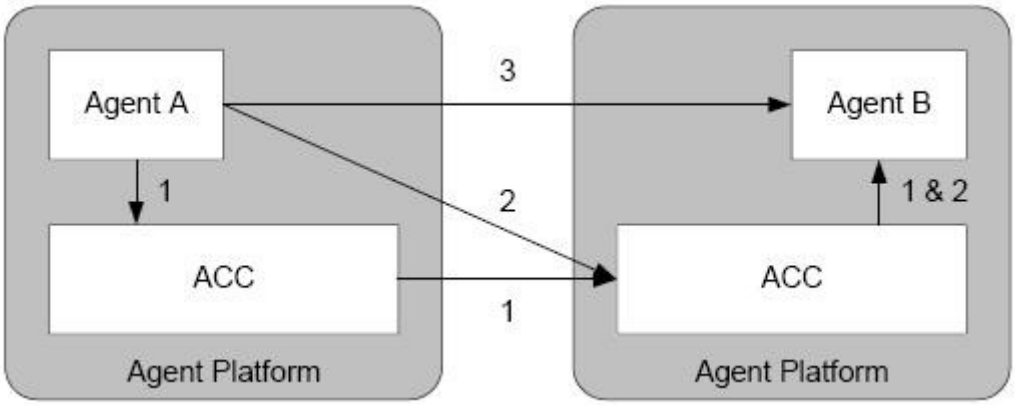
FIPA vermeldt dat de transportdienst door een Agent Communication Channel (ACC) verzorgd wordt. Dit kanaal dat eigen is aan een MTP zorgt voor het transport van de 'payload' tussen agenten op een systeem. Om deze dienst uit te voeren kan de ACC beschikken over de informatie van de AMS en de DF. Om berichten via het MTP te versturen naar een andere ACC dient de 'payload' omkapseld te worden met een enveloppe. Dit mechanisme is in bovenstaande figuur 9 weergegeven. De ontologie voor een enveloppe te beschrijven die opgenomen is ter illustratie in figuur 10 maakt deel uit van de `fipa-agent-management` ontologie. Een enveloppe is een verzameling van parameters die minstens de `to`, `from`, `date` en `acl-representation` parameters omhelst. Ter herinnering kunnen we nog vermelden dat een `AID` een agent kenmerkt zodat er in de `to` en `from` velden dus een `AID` dient te staan.

Frame Ontology	envelope fipa-agent-management			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
to	This contains the names of the primary recipients of the message.	Mandatory	Sequence of agent-identifier	
from	This is the name of the agent who actually sent the message.	Mandatory	agent-identifier	
comments	This is a comment in the message envelope.	Optional	string	
acl-representation	This is the name of the syntax representation of the message payload.	Mandatory	string	fipa.acl.rep.biteffic ient.std fipa.acl.rep.string.s td fipa.acl.rep.xml.std
payload-length	This contains the length in bytes of the message payload.	Optional	string	
payload-encoding	This contains the language encoding of the message payload.	Optional[5]	string	US-ASCII ISO-8859-1 … ISO-8859-9 UTF-8 Shift_JIS EUC-JP ISO-2022-JP ISO-2022-JP-2
date	This contains the creation date and time of the message envelope	Mandatory	date	
intended-receiver	This is the name of the agents to whom this instance of a message is to be delivered.	Optional	Sequence of agent-identifier	
received	This is a stamp representing the receipt of a message by an ACC.	Optional	received-object	
transport-behaviour	This contains the transport requirements of the message.	Optional	(Undefined)[6]	

Figuur 10: Definiëring enveloppe in de agent-management ontologie (FIPA)

Het formaat van een enveloppe kan dan weer in verschillende vormen voorkomen zodat de volgende twee specificaties ontwikkeld zijn: FIPA Agent Message Transport Envelope Representation in XML Specification [00085] en FIPA Agent Message Transport Envelope Representation in Bit Efficient Specification [00088].

Een ACL-bericht kan door een agent op slechts twee manieren verstuurd worden. Een agent kan het bericht naar zijn eigen ACC sturen via de interface die standaard aanwezig is voor het platform waarop hij verblijft. De ACC zal dan vervolgens via een geschikt transportprotocol het adres verder sturen naar de 'remote' ACC die vervolgens het bericht doorstuurt naar de geadresseerde agent (optie 1 in figuur 11). Een tweede optie (optie 2 in figuur 11) is dat de agent de 'payload' direct naar de 'remote' ACC stuurt. Het is hierbij wel vereist dat de agent de inferface van de ACC van dit platform kent. Optie 3 op figuur 11 wordt niet ondersteund door FIPA.



Figuur 11: Mogelijke paden van een bericht (FIPA)

De specificaties van een transport van een AP zijn opgenomen in de `fipa-agent-management` ontologie en zijn uitgedrukt in de `fipa-sl0`. In onderstaande frames is de ontologie terug te vinden om de transporteigenschappen van een AP te beschrijven. In figuur 12 zien we de beschrijving van het AP zelf met parameters voor de naam (`Name`) en de diensten op het AP (`Ap-services`). Vervolgens zijn in figuur 13 de parameters gegeven voor de beschrijving van dergelijke diensten op een AP, namelijk `Name`, `Type` en `adresses`.

Frame Ontology	`ap-description` `fipa-agent-management`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`Name`	The name of the AP.	Mandatory	`string`	
`Ap-services`	The set of services provided by this AP to the resident agents.	Optional	Set of `ap-service`	

Figuur 12: Definiëring AP in de agent-management ontologie (FIPA)

Frame Ontology	`ap-service` `fipa-agent-management`			
Parameter	**Description**	**Presence**	**Type**	**Reserved Values**
`Name`	The name of the AP Service.	Mandatory	`string`	
`Type`	The type of the AP Service.	Mandatory	`string`	`fipa.mtp.*`
`addresses`	A list of the addresses of the service.	Mandatory	Sequence of `url`	

Figuur 13: Definiëring diensten op een AP in de agent-management ontologie (FIPA)

Om over het internet te reizen, dienen berichten een applicatiestandaard en transportprotocol te gebruiken. De applicatielaag is de bovenste laag in de gelaagde internetstructuur. De bekendste standaard die FIPA ondersteunt, is wellicht HTTP. Toch is

het IIOP het standaard protocol voor berichtuitwisseling volgens FIPA. In de FIPA Agent Message Transport Protocol for HTTP Specification [00084] en FIPA Agent Message Transport Protocol for IIOP Specification [00075] wordt beschreven hoe FIPA-berichten (payload en enveloppe) via internet van de ene agent naar de andere gestuurd kunnen worden.

4.6.5 Interactieprotocollen

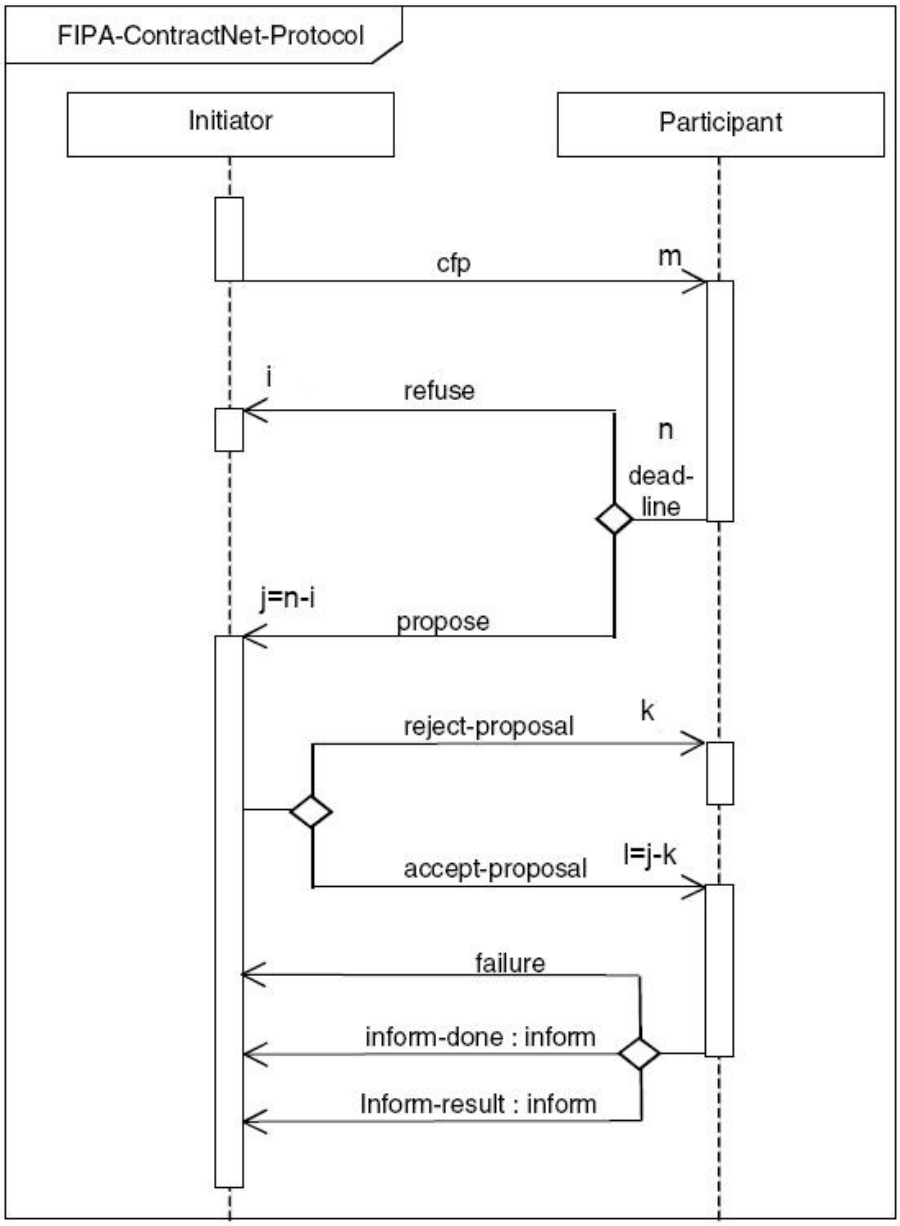
Sommige berichtuitwisselingen tussen agenten zijn onderling aan elkaar gelinkt zodat ze samen een conversatie vormen. Deze complexe berichtuitwisselingen worden door FIPA ondersteund door specificaties te geven aan abstracte interacties die geconcretiseerd kunnen worden door programmeurs tot ware conversaties.

Tabel 6: Interactieprotocollen met het FIPA identificatienummer

• Request	fipa [00026]
• Query	fipa [00027]
• Request When	fipa [00028]
• Contract Net	fipa [00029]
• Iterated Contract Net	fipa [00030]
• Brokering	fipa [00033]
• Recruiting	fipa [00034]
• Subscribe	fipa [00035]
• Propose	fipa [00036]

Elk interactieprotocol in bovenstaande tabel 6 standaardiseert de uitwisseling van ACL-berichten tussen een initiërende en een of meerdere participerende agenten. Een bepaald protocol wordt geïdentificeerd door de naam van het gebruikte protocol in het `:protocol` slot van het ACL-bericht te vermelden. Dit model van uitwisseling van ACL-berichten specificeert de opeenvolging van 'communicative acts' die kunnen plaatsvinden tussen beide agenten. Vervolgens definieert elk document de 'exceptions' die kunnen optreden bij een bepaald protocol en hoe men deze kan opvangen. De gekozen namen van de protocollen zijn intuïtief duidelijk maar bij wijze van voorbeeld behandelen we toch het Contract Net protocol.

Via het Contract Net protocol kunnen we conversaties tussen agenten opzetten waar de initiator een taak uitgevoerd wil hebben. Bij de uitvoering van deze taak wil hij een bepaalde parameter die afhankelijk is van de uitvoering optimaliseren. Een voorbeeld dat trouwens goed aansluit bij deze thesis kan zijn dat een agent een transportvraag wil laten invullen door de agenten die de transporteurs vertegenwoordigen. Afhankelijk van de overwegingen zal de agent dan proberen de prijs, stiptheid of andere parameters te optimaliseren. De flow van berichten kan als volgt grafisch voorgesteld worden:



Figuur 14: Berichtenstroom bij het FIPA-ContractNet-Protocol (FIPA)

Zoals te zien is op figuur 14 gaat het protocol van start als de initiator een `Call for Proposal` stuurt aan `m` andere agenten waarvan hij denkt dat deze de taak kunnen invullen. Deze `Call for Proposal` is één de 'communicative acts' gedefinieerd door FIPA uit tabel 5.

Het volgende voorbeeld wordt in het FIPA-document aangesneden en wil zeggen dat agent `j` agent `i` uitnodigt om een voorstel te doen betreffende de prijs als hij `50` 'boxes' van het voorwerp met label 'plum' wil kopen. Een bod is enkel geldig als de prijs kleiner is dan '10' eenheden. Het label 'plum' is opgenomen in de 'fruit-market ontology'.

```
(cfp
:sender (agent-identifier :name j)
:receiver (set (agent-identifier :name i))
:content
            "((action (agent-identifier :name i)
            (sell plum 50))
            (any ?x (and (= (price plum) ?x) (< ?x 10))))"
:ontology fruit-market
:language fipa-sl)
```

Na een bepaald tijdsinterval (deadline) hebben `n` agenten het `CFP` bericht ontvangen. Van deze `n` agenten zullen `i` agenten de taak niet kunnen of willen uitvoeren. Zij sturen daarom een `refuse` bericht terug. De andere `j`, meer bepaald `n-i`, agenten zijn wel geïnteresseerd om de taak uit te voeren en doen een voorstel betreffende de parameter opgegeven door de initiator-agent. Dit doen ze via de 'communicative act' `propose`. Volgend voorbeeld is een verderzetting van het voorbeeldje bij de `CFP`. Agent `j` doet een voorstel aan agent `i` om `50` eenheden van 'plum' te verkopen aan een prijs van `5` eenheden.

```
(propose
:sender (agent-identifier :name j)
:receiver (set (agent-identifier :name i))
:content
            "((action j (sell plum 50))
            (= (any ?x (and (= (price plum) ?x) (< ?x 10))) 5)"
:ontology fruit-market
:in-reply-to proposal2
:language fipa-sl)
```

Na het ontvangen van dit bericht zal de initiator-agent dit bericht aanvaarden of verwerpen door respectievelijk een `reject proposal` en `accept-proposal` te versturen.

Na het ontvangen van een acceptatiebericht zal de leverende agent de taak uitvoeren. Na het uitvoeren van de taak licht de uitvoerende agent de vragende agent in over de uitvoering van de taak.

Verder zijn er voor dit standaarddocument van het Contract Net protocol nog uitzonderingen geformuleerd. Deze uitzonderingen tonen ons hoe agenten reageren als er bepaalde berichten verstuurd worden die zij niet begrijpen. Het kan hier gaan om een misopvatting van de ontologie, de specificaties van het protocol of de 'content' taal.

4.7 Actuele bezigheden van FIPA en toekomstvisie

4.7.1 Content Languages

FIPA werkt momenteel aan een drietal 'content languages' als opvolger en aanvulsel op hun `fipa-sl` taal. Op deze manier hoopt men een breder gamma aan syntactische mogelijkheden aan te bieden aan programmeurs om zo de inhoud en mogelijkheden van de ACL-berichten te verrijken.

Ten eerste heeft men de KIF Content Language. Via deze taal wil men ontwikkelaars de mogelijkheid geven hun ACL content parameter te specificeren met een reeds bestaande standaard namelijk de American National Standard (ANSkif) for Knowledge Interchange Format (KIF)[XII]. Andere talen zoals de CCL en RDF talen zijn er op gericht om via XML-schema's meer flexibiliteit en mogelijkheden te geven om de content van ACL-berichten mee uit te drukken.

4.7.2 Ondersteuning voor meerdere heterogene interfaces

FIPA probeert om voor één dienst meerdere specificaties te ontwikkelen. Op deze manier ontwikkelen zij meerdere interfaces waarlangs agentplatformen of agenten zelf interoperabel kunnen zijn. Poslad et al. (2001) vertellen over deze manier van werken: "[…] a move away from supporting single external interfaces towards supporting multiple external interfaces thus easing the maintenance of heterogeneous external interfaces".

[XII] Deze inhoudstaal wordt tevens gebruikt door KQML als inhoudstaal.

Een voorbeeld hiervan is dat de interfaces voor oorspronkelijke transportprotocollen IIOP en HTTP aangevuld zullen worden met het Wireless Application Protocol (WAP).

4.7.3 Overige

Verder werkt FIPA nog aan de standaardisatie van nieuwe interactieprotocollen om de mogelijkheden tot conversaties uit te breiden. Een ander interessant domein waar FIPA zich op concentreert, is het ontwikkelen van specificaties van een nieuwe dienst als onderdeel van het platform. Zo is er de FIPA Ontology Service Specification [00086] die een Ontology Agent voorziet waar men ontologieën kan registeren en gebruiken om de communicatie te verzorgen. De inspanningen op het gebied van Agent Discovery liggen ook in de lijn met de verwachtingen. Deze processen om dynamisch diensten en agenten te detecteren op een systeem zijn immers nog niet expliciet behandeld in de standaarden. Een exhaustieve lijst van nieuwe specificaties waar FIPA aan werkt is te vinden in bijlage III.

4.8 Kritieken op FIPA

De meeste betrokkenen in de agentwereld beschouwen de FIPA-ACL als dé standaard ACL. De standaardisering van de interacties op het niveau van de communicatie tussen agenten heeft dus zijn doel bereikt. Dit wordt bevestigd in mijn interview met dr. ir. Roos. Toch is de ontologie die gebruikt wordt in een ACL vaak zeker zo belangrijk als de ACL zelf. Zoals we besproken hebben, is er nog geen standaardspecificatie voor het management van ontologieën uitgevaardigd. Hoewel FIPA ontologieën specificeert, zijn deze eigen aan het beheer en het gebruik van agenttoepassingen. Een voorbeeld hiervan is de ontologie voor agentmanagement. De specificatie voor het gebruik van ontologieën om inhoud mee te geven aan berichten is slechts een experimentele specificatie zodat volledige interoperabiliteit op basis van FIPA-standaarden niet mogelijk is. Ontwikkelaars zullen het immers eens moeten geraken over de te gebruiken ontologie voor het construeren van de inhoud van de uit te wisselen berichten.

Betreffende de standaardisering van de componenten en het gebruik van software-infrastructuur in agenttoepassingen kunnen we ook enkele kritieken formuleren. Singh

(1998) vermeldt immers dat de hoge mate van formele specificaties voor het AP niet geschikt is voor alle agenttoepassingen. Het interview met prof. Holvoet bevestigt deze opvatting. Zeker voor toepassingen in de industrie zijn de vereisten vaak zodanig specifiek dat algemene standaarden zoals FIPA eerder een beperking vormen dan een hulp.

Verder wordt niet alleen FIPA maar ook de agentgemeenschap in het algemeen een te enthousiaste aanpak verweten. De standaarden voor de opbouw van agentplatformen proberen immers interoperabiliteit na te streven tussen onafhankelijke software-entiteiten, in dit geval agenten. (Omair et al., g.d.; Dickinson, 2004) Deze beweging naar interoperabiliteit gebeurde eveneens met het idee van agenten als een soort van 'wrapper' of omhulsel voor systemen met geen agentkenmerken. Op deze manier zou men dan ook integratie van toepassingen zonder agentkenmerken kunnen integreren door een agent als soort van spreekbuis te gebruiken. Deze evoluties gebeurden op een moment dat er betreffende de opbouw van agenten en agentsystemen zelf nog veel vraagstukken en onduidelijkheden waren. (Hendler et al., 2007)

Verder zijn er enkele standaarden in de industrie die ook de integratie van onafhankelijke software-entiteiten via het Web probeerden te bereiken. Deze standaarden betreffen hoofdzakelijk Web Services en Grid Computing en kunnen rekenen op bijval door grote industriële spelers. Op deze manier vinden we deze technologieën meer en meer terug in toepassingen. In die zin lijkt het erop dat FIPA en andere organisaties die interoperabiliteit van softwareagenten probeerden te bewerkstelligen voorbij gestoken worden door de pas vernoemde technologieën. (Omair et al., g.d.) Ook de geïnterviewde personen formuleerden hun bedenkingen bij de manier waarop agentsystemen gerealiseerd zullen worden. Hoewel de toekomst niet echt duidelijk is, vrezen ook zij dat andere standaarden de bovenhand zullen halen op de FIPA-standaarden.

Het is natuurlijk zo dat de aangehaalde standaarden een goede basisinfrastructuur vormen voor agenttoepassingen. Veelbelovende artikels en papers zijn dan ook terug te vinden die de infrastructuur die opgebouwd kan worden met behulp van Web Services en Grid Computing kan verrijken met de kwaliteiten van agenten. Ook FIPA ziet in dat vooral de opkomst van Web Services een belangrijke evolutie kan zijn voor agenttoepassingen (FIPA Website, 2007c). Het bewijs hiervan zijn de inspanningen van de gevormde IEEE-FIPA groep die voor AAMAS 2007 een paper over de integratie van

FIPA en Web Services publiceert. Verder wordt er vermeld dat de integratie van Web Services in FIPA gestandaardiseerd zal worden in een nieuwe FIPA-specificatie. (FIPA Website, 2007b)

HOOFDSTUK 5: Agent-platformen of -middleware

5.1 Middleware

Middleware wordt door Laudon en Laudon (2005) gedefinieerd als: "Software that connects two disparate applications, enabling them to communicate with each other and to exchange data". Deze software is speciaal ontwikkeld om een interface tussen twee applicaties te vormen. Functionaliteit die mogelijk gemaakt wordt door een softwaretoepassing kan dus via deze interface eenvoudig gebruikt worden in een andere toepassing. Ook de bekende bibliotheekklassen van verschillende programmeertalen, met Java op kop, kan gerekend worden bij deze middleware. Door het objectgeoriënteerde paradigma kan men immers logisch samenhangende data zoals variabelen en methoden samen gaan inkapselen in een klasse. Deze klasse kan men dan ter beschikking stellen van andere programmeurs via een Application Program Interface (API). Middleware bezit vaak functionaliteit die niet gerelateerd is aan de kern van de applicatie maar hier los van staat. Zo bezit het vaak de mogelijkheden voor datatoegang, versleutelen en ontsleutelen van gegevens, communicatie tussen systeemcomponenten, netwerk overwegingen, basis-analyses en -functionaliteit en controle van systeemmiddelen (Laudon en Laudon, 2005). In dit hoofdstuk wordt ingegaan op de waarde die middlewareplatformen kunnen betekenen voor agenttechnologie.

5.2 Nut van agent middleware

5.2.1 Toepassing van het paradigma

In een voorgaand hoofdstuk hebben we kunnen zien in welke rollen agentapplicaties zich kunnen bevinden en over welke eigenschappen ze in vele gevallen bezitten. De interne structuur van een agentschap hebben we een 'agency' genoemd en is meestal een BDI of sociale structuur. Agenten kunnen deelnemen aan onderhandelingen en conversaties en

nemen autonoom beslissingen. Deze agenten worden ontwikkeld via het agent paradigma. Om een dergelijk systeem naar behoren te ontwikkelen, moet een programmeur een gedegen kennis hebben van dit paradigma. Bij Poggi et al.(2002) vinden we het volgende citaat: "[…] developers trying to use the new technology not only need to know its main principles (e.g. ACL, ontologies, and interaction protocols), but they also need to fully understand their implications on how the software is to be written, their interaction with the existing programming languages and operating environments, and so on." Dit is natuurlijk een erg grote drempel voor programmeurs zodat het adoptieproces van agenttechnologie belemmerd wordt.

Wanneer men middleware gebruikt, zitten de grondslagen van het paradigma al vervat in de klassen van een bibliotheek. Programmeurs kunnen via de APIs gemakkelijk deze functionaliteit correct en naar de normen van het paradigma gebruiken in hun agentsysteem. Het grote voordeel is dat ze zich kunnen concentreren op de ontwikkeling van de businesslogica en niet de kenmerken en vereisten van het paradigma tot in detail dienen te kennen. (Poggi et al., 2002; Omicini en Rimassa, 2004)

5.2.2 Snellere adoptie

Doordat een middleware-platform tot op zekere hoogte de abstracte concepten concretiseert die eigen zijn aan een bepaald paradigma, zorgt het voor een verhoogde kans op acceptatie door de markt. Deze makkelijk bruikbare code zorgt immers voor een lagere barrière om toepassingen te ontwikkelen die gebruik maken van een bepaalde technologie. Programmeurs gaan dus ervaring opdoen met de geïmplementeerde technologie van het platform en interessante facetten of opportuniteiten ervan blootleggen. Poggi et al. (2002) vermeldt dat hier een mogelijkheid tot serendipiteit kan ontstaan. Men kan bepaalde voordelen ontdekken waar men initieel niet naar op zoek was door de technologie die vervat is in het platform te gebruiken.

5.2.3 Ontwikkelingsomgeving

Vermits agenttechnologie een nieuw paradigma is, is men ook beginnen sleutelen aan de methode om projecten te plannen en te implementeren. We gaan in deze eindverhandeling niet verder in op deze methoden maar een agent-framework kan ook

Computer Aided Software-Engineering (CASE) tools aanbieden die de ontwikkelaar helpen met zijn werkzaamheden. (Bellifemine, 2003)

5.2.4 Bibliotheken met functionaliteit

Agenttechnologie is een technologie die veelvuldig gebruik maakt van ondersteunende software-infrastructuren en -concepten. Deze software-infrastructuren maken onder andere dynamische, componentgebaseerde en gedistribueerde systemen mogelijk. Deze verschuiving is bewerkstelligd door basistechnologieën zoals XML en RDF maar ook door ontwikkelingen op e-business gebied waar standaarden als ebXML en RosettaNet ontwikkeld zijn. Verder zijn er steevast verschuivingen gekomen in hoe machines samenwerken door UPnP, Jini en andere samenwerkingsmechanismen. Een andere en waarschijnlijk dé belangrijkste infrastructuur die stilaan van de grond komt zijn Web Services. Tot slot zijn verschillende concepten stilaan doorgedrongen die ervoor zorgen dat de IT-wereld vanuit een ander perspectief bekeken kan worden. We praten hier dan over concepten zoals Semantic Web, Service Oriented Architecture, P2P computing, Grid Computing en Autonomous Computing. (AgentLink III, 2004) Enkele belangrijke evoluties in software en toepassingsarchitecturen worden als afsluiting van dit hoofdstuk beschreven in punt 5.3.

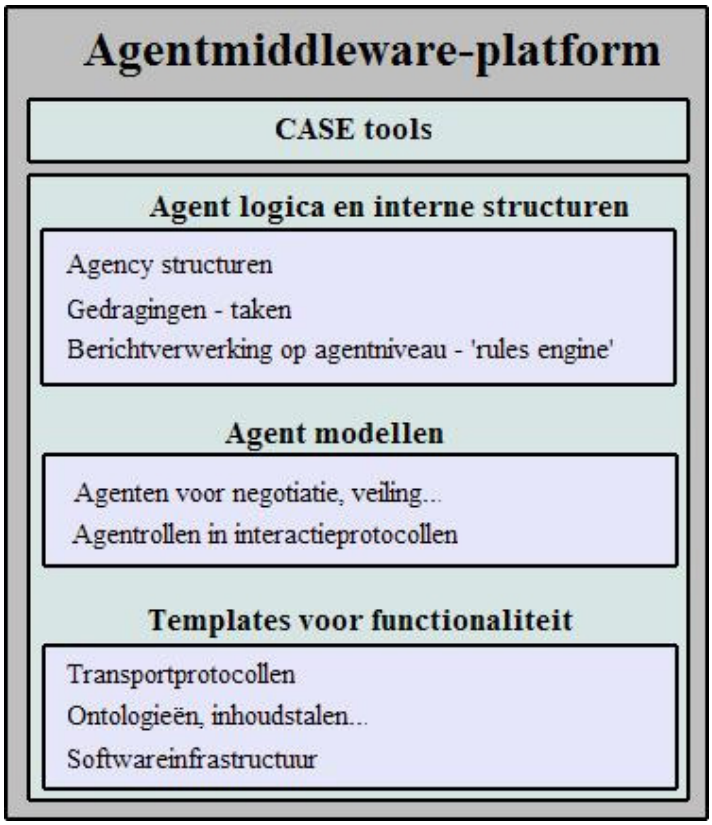
Bovendien hebben we gezien dat er binnen een agentsysteem systeemtaken moeten voorzien worden zodat agenten actief kunnen zijn op een platform. Via deze componenten op systeemniveau worden agenten ondersteund in hun processen. We herinneren ons uit hoofdstuk 3 wat een agent is en moet kunnen en welke functies multi-agentsystemen aanbieden om dit te ondersteunen.

Naast bibliotheekklassen met abstracte interfaces die gebruikt kunnen worden opdat het paradigma van agenttechnologie gerespecteerd wordt, kunnen ook meer concrete klassen voorzien worden. Deze klassen zijn agenten waarvan de ‘agency’, de taken en de manier van interageren reeds gecodeerd zijn zodat ze in feite dadelijk bruikbaar zijn in een agenttoepassing. Deze agenten kunnen meestal ingezet worden voor specifieke functies binnen een multi-agentsysteem. Vaak voeren dergelijke agenten een functie uit die in vele applicaties nuttig kan zijn zoals het voeren van negotiaties of het vertegenwoordigen van een bedrijf of persoon in een online veiling maar ook het

bekleden van bepaalde rollen van agenten in interacties. Een voorbeeld hiervan is de initiator van een Contract Net protocol.

Verder kunnen klassen voorzien worden voor de componenten, vaak ook agenten, die nodig zijn op systeemniveau. De functionaliteit op systeemniveau kan onafhankelijk geïmplementeerd worden van de agenten zelf. Betreffende de functionaliteit zijn deze elementen immers identiek in verschillende agentsystemen. Het is daarom dat men via middleware-platformen efficiënter kan ontwikkelen omdat functies zoals bijvoorbeeld agentmanagement- en communicatiemogelijkheden reeds voorzien zijn. De programmeur hoeft zich als het ware enkel bezig te houden met het implementeren van de businesslogica en niet met het ontwikkelen van ondersteunende functies.

Samen met de voordelen van agentmiddleware in voorgaande alinea's kan men dus besluiten dat men via agentplatformen niet alleen efficiënter maar ook effectiever agenttoepassingen kan ontwerpen. In onderstaande afbeelding vinden we nog eens een overzicht terug van agentplatformen en hun functionaliteit.



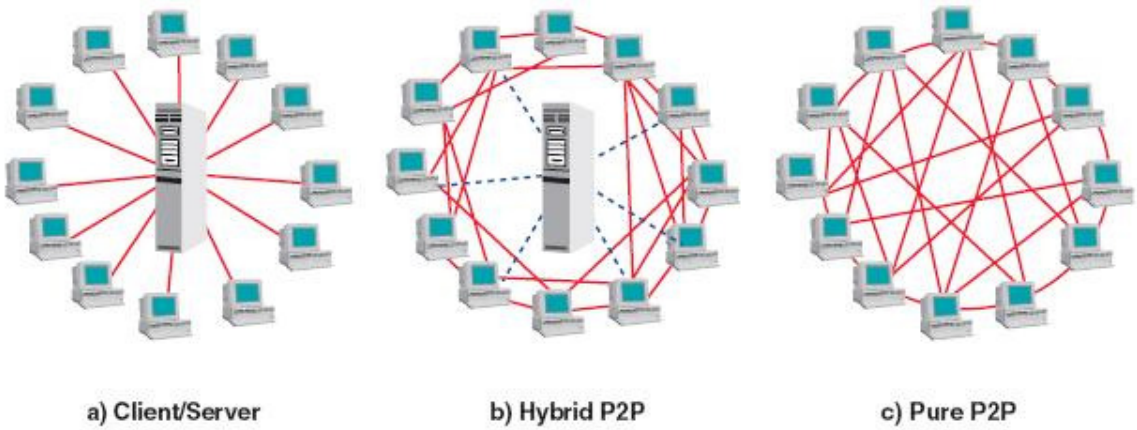
Figuur 15: Overzicht agentmiddleware-platform

5.3 Ondersteunende technologieën voor multi-agentsystemen

5.3.1 Peer-to-peer networks

Een peer-to-peer toepassingsarchitectuur bestaat uit een netwerk van gelijken of peers. De functionaliteit van een applicatie wordt dan voorzien door het al dan niet dynamisch samenwerken van verschillende van deze peers (Panko, 2005). Deze applicaties bevinden zich op verschillende gebruikerscomputers en vormen samen een gedistribueerd netwerk en kenmerken zich door de onderlinge gelijkheid. Deze gelijkheid slaat op het afwezig zijn van een kenmerkende hiërarchie tussen de entiteiten, ook wel nodes genoemd.

Toch kunnen in sommige gevallen servers instaan voor het aanbieden van faciliterende taken die best gecentraliseerd worden. Op deze manier bekomen we een hybride systeemarchitectuur. De servers voorzien dan vaak een centrale index die instaat voor het opzoeken en terugvinden van bijvoorbeeld ingelogde peers en diensten die zij aanbieden (Bellifemine, 2003). In tegenstelling tot meer traditionele client/server-toepassingen zijn peer-to-peer (P2P) toepassingen interessanter in het licht van robuustheid, schaalbaarheid, onderhoud en de ontplooiing van het systeem zelf (Luck et al., 2005). De drie verschillende applicatiearchitecturen zijn terug te vinden in onderstaande tekening.



Figuur 16: Verschillende applicatiearchitecturen (Bellifemine, 2003)

Het hoeft geen betoog dat de besproken P2P-applicatiearchitectuur uitstekend samengaat met het agentparadigma. Agentapplicaties kunnen gecentraliseerd worden op

één machine maar zijn in werkelijkheid altijd gedistribueerde applicaties. Verder dienen P2P-applicaties over protocollen te beschikken voor informatie-uitwisseling en lokalisatie van entiteiten of diensten binnen het systeem. Ook dergelijke diensten vinden we terug op systeemniveau van een multi-agentsysteem (Luck et al., 2005).

5.3.2 Java

Java is een objectgeoriënteerde programmeertaal die in de jaren '90 ontwikkeld werd door een dochterbedrijfje van Sun Microsystems. Door het feit dat Java objectgeoriënteerd is, leent deze taal zich tot het ontwikkelen van agentsystemen. We hebben immers reeds gezien dat de verschillen tussen agenten en objecten eerder klein zijn en dat een agent een meer gesofisticeerd object is. De grote kracht en populariteit van Java is de makkelijkheid waarmee klassen kunnen herbruikt worden. Op deze manier is de Java-syntax altijd erg eenvoudig gebleven omdat bijkomende functionaliteit in de Java bibliotheek ter beschikking komt te staan in de vorm van nieuwe klassen. Programmeurs kunnen klassen uit de bibliotheek gebruiken of overerven en methodes overschrijven om zo snel de gewenste functionaliteit te verkrijgen. Het grote voordeel hiervan is dat men niet de werking van elke klasse dient te kennen maar gewoon de functionaliteit van de ontwikkelde klasse hoeft te weten om ze te gebruiken.

Toch is Java veel meer dan een programmeertaal en bibliotheek alleen. De Java bibliotheek is slechts een onderdeel van een groter Java-platform dat een verzameling van programma's en klassenbibliotheken is. Hoewel een bespreking van het volledige Java-platform buiten het bestek van deze thesis valt, wil ik toch enkele elementen verder toelichten omdat ze belangrijk kunnen zijn bij het ontwikkelen van een multi-agentsysteem.

Zo zorgt de Java Virtual Machine (JVM) ervoor dat Java platformonafhankelijk uitvoerbaar is. Eerst wordt Java broncode door een compiler gecompileerd tot bytecode. Deze bytecode wordt door de JVM van het toestel waarop de code uitgevoerd wordt, vertaald naar instructies die uitvoerbaar zijn door de processor van het toestel. Op deze manier

kan dezelfde Java bytecode uitgevoerd worden op verschillende platformen[XIII]. Dit wordt ook wel de platformonafhankelijkheid van Java genoemd. Deze JVM is een onderdeel van een breder programma-pakket ook wel Java runtime environment genoemd (JRE).

Van het Java-platform bestaan er ondertussen drie versies, namelijk: Java 2 Standard Edition (J2SE), Java 2 Enterprise Edition (J2EE) en Java 2 Micro Edition (J2ME). Elk van deze platformen heeft een andere invalshoek. Zo wordt J2SE aanzien als het standaardpakket voor het ontwikkelen van applicaties waarop J2EE een extensie is met gespecialiseerde API's voor onder andere serverapplicaties. J2ME is dan weer speciaal ontwikkeld om rekening te houden met de beperkte rekencapaciteit van mobiele toestellen zoals PDA's. (Venners, 1996)

5.3.3 Java RMI en CORBA

Integratie van verschillende systemen is altijd een belangrijk agendapunt geweest binnen de IT-wereld. De technologie die Object-orientated Remote Procedure Call (ORPC) genoemd werd, probeerde een antwoord te zijn op deze vraag naar integratie. Deze technologie zorgde ervoor dat men via objecten in een bepaalde markeertaal methoden van externe applicaties kon oproepen. Deze objecten zijn in feite eenvoudige kopies van de applicaties waartussen men methodeoproepen wil uitwisselen en worden ook wel 'stubs' genoemd. De integratie van de toepassingen is mogelijk doordat deze objecten de interne complexiteit van de programma's verbergt. Deze methodeaanroepen worden getransporteerd tussen de 'stud' van de cliënt en de server via een Remote Procedure Call (RPC) protocol. Deze technologie mondde uit in enkele concrete platformen zoals CORBA en Java RMI. (Depaire, 2003; Delsupehe, 2005)

CORBA is een specificatie van een ORPC dat ontworpen is om gedistribueerde softwaretoepassingen te laten samenwerken. Deze toepassingen kunnen ontwikkeld zijn in verschillende talen zoals daar zijn C, C++, Java, Ada, en Smalltalk. Verder kunnen CORBA-objecten bestaan op diverse systemen zoals onder andere Solaris, Windows 95/NT, OpenVMS, Digital Unix, HP-UX, en AIX. (Morgan, 1997)

[XIII] Met platform wordt bedoeld het geheel van infrastructuur waarop een applicatie dient te werken zoals het besturingssysteem en de hardware. Dit concept mag dus niet vereenzelvigd worden met een middleware-platform.

Figuur 17: De CORBA-structuur

In bovenstaande afbeelding kunnen we de CORBA-architectuur aanschouwen. CORBA-diensten worden beschreven door een interface in de OMG Interface Definition Language (IDL). Op deze manier wordt er een 'client stub' en 'server stub' gecreëerd die de complexiteit van de betreffende client- en serverapplicaties voor elkaar afschermen. Via de CORBA-technologie kunnen objecten methoden aanroepen van 'remote' objecten en wordt het mogelijk gemaakt om data te transporteren tussen deze twee objecten. Alle oproepen passeren via de Object Request Broker (ORB) die instaat voor de vertaling van dataformaten en voor het lokaliseren van de CORBA-diensten. Deze ORBs communiceren via het Internet Inter-ORB Protocol (IIOP). (Reilly, 2006; Morgan, 1997)

Java Remote Method Invocation (Java RMI) zorgt ervoor dat de programmeur gedistribueerde Java-applicaties kan ontwikkelen. Via Java RMI kan men methodes van Java-objecten aanroepen die op een andere JVM kunnen draaien. Op deze manier ondersteunt het Suns beroemde citaat: "Write Once, Run Anywhere model". Deze functionaliteit bevindt zich in de Java bibliotheek in het package `java.rmi`. Via RMI kunnen volledige objecten als parameters verstuurd worden. Dit staat in schril contrast met andere RPCs die enkel primitieve parameters of samenstelling van primitieve parameters kunnen uitwisselen. In een 'RMI registry' kan een client-applicatie zoeken naar 'RMI services'. Als deze services nieuwe klassen vereisen worden deze afgehaald van een webserver door de client-applicatie. (Sun Website, 2007; Reilly, 2006)

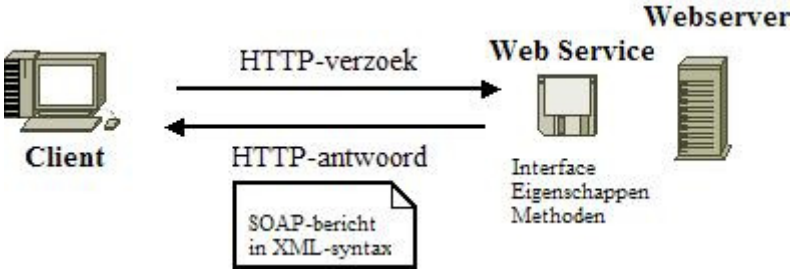
5.3.4 Web Services

Web Services is een relatief nieuwe aanpak om heterogene componenten te laten samenwerken via internettechnologie. Hiermee is het belangrijk dat deze samenwerking

onafhankelijk is van interne datastructuren, gebruikte programmeertalen waarin de toepassingen gecodeerd zijn of systemen waarop de toepassingen draaien. (Depaire, 2003) Het is een verbetering ten opzichte van technologieën als CORBA en Java RMI omdat het een globale standaard is die de mogelijkheid biedt om alle software-componenten te integreren onafhankelijk van hun interne structuur. Bij CORBA en Java RMI daarentegen moesten de componenten die men wou integreren wel aan bepaalde voorwaarden voldoen. De belangrijkste componenten van Web Services zijn overzichtelijk gedefinieerd in Noordzij (g.d.) als volgt:

- *Simple Object Access Protocol (SOAP), de basistechniek om functionaliteit aanroepbaar te maken.*
- *Web Service Definition Language (WSDL), de definitie taal om de aangeboden diensten te beschrijven*
- *Universal Description, Discovery and Integration (UDDI), het mechanisme om diensten op een centrale locatie vast te leggen en vindbaar te maken voor gebruikers.*

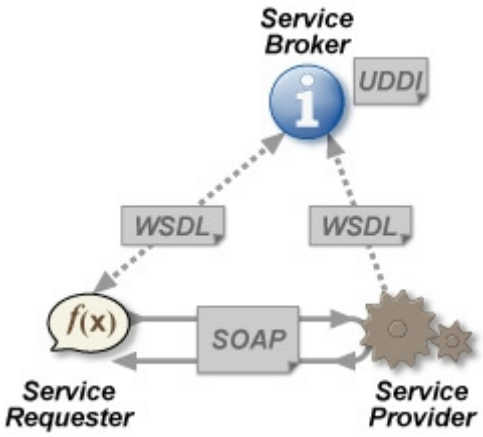
SOAP is een berichtformaat waarin (web) clients instructies naar webdiensten kunnen versturen om methodes aan te roepen die door de desbetreffende diensten worden ondersteund. SOAP is ontwikkeld om het uitwisselen van informatie in gedistribueerde omgevingen te standaardiseren en te vereenvoudigen. Het is een protocol dat XML gebruikt om het berichtformaat te beschrijven. Daarnaast maakt het gebruik van HTTP, het standaard Internet communicatie protocol, om de berichten tussen systemen te versturen. (Panko, 2005; Noordzij, g.d.) Een illustratie van deze interactie vinden we terug in onderstaande afbeelding.



Figuur 18: Interactie client en Web Service (eigen verwerking van Panko, 2005)

In voorgaande paragraaf hebben we besproken hoe applicaties 'ontsloten' kunnen worden en gebruikt kunnen worden als Web Service via SOAP. Het is nu zaak om de

dienst zodanig te beschrijven dat deze ook daadwerkelijk gebruikt kan gaan worden. Hiervoor is de 'Web Service Definition Language' (WSDL) ontwikkeld. Een op XML gebaseerde taal die beschrijft hoe de functie aangeroepen moet worden, welke gegevens door de dienst verwacht worden, wat het transportmedium is en hoe het antwoord gestructureerd is. WSDL wordt ook gebruikt om een 'proxy' te genereren. Deze 'proxy' is vergelijkbaar met een 'stub' uit het CORBA-gebeuren en is in feite een kopie van de Web Service. De beschrijvingen van de Web Services worden vervolgens kenbaar en toegankelijk gemaakt via Universal Description, Discovery and Integration-protocol (UDDI). Op deze manier vormt UDDI een 'yellow-pages'-dienst waar Web Services geregistreerd worden en gezocht kunnen worden. Zoals in onderstaande figuur weergegeven is kunnen we dus drie verschillende interacties terugvinden. In eerste plaats gaat de aanbieder van de Web Service deze registreren in het UDDI-register. Vervolgens kan degene die de Service vraagt zoeken naar bepaalde diensten het UDDI-register. Tot slot zal degene die nood heeft aan de Service HTTP en SOAP berichten uitwisselen met de Web Service zelf.



Figuur 19: Mechanismes bij een Web Service (Wikipedia, 2007)

5.3.5 Grid Computing

Naar de Grid wordt verwezen als "the high-performance computing infrastructure for supporting large-scale distributed scientific endeavour". (Agentlink III, 2004) Op deze manier is het mogelijk om computers, netwerken, databases op een geïntegreerde en samenwerkende manier te gebruiken. Op deze manier worden processen waar enorme rekenkracht voor nodig is en die een enorme hoeveelheid data nodig hebben snel

verwerkt doordat alle apparaten hun verwerkingsmiddelen delen. (Panko, 2005) De Grid is eigenlijk een P2P-framework dat de mogelijkheid biedt om de rekenkracht van gedistribueerde processoren te benutten. Het bekendste voorbeeld hiervan is het SETI@home-project[XIV] waar thuisgebruikers een kleine toepassing dienen te installeren op hun PC om in rust processorcapaciteit ter beschikking te stellen voor de analyse van data. De Grid speelt op deze manier in op de tekortkomingen op dit gebied van huidige internettechnologie en Web-infrastructuur.

5.3.6 Semantic Web

Het Semantic Web biedt een manier om informatie te organiseren zodat het interpreteerbaar wordt voor machines. Dit kan bewerkstelligd worden door data op het web op een semantisch niveau te beschrijven en onderling te linken. Het semantische web kan gezien worden als een extensie van het World Wide Web waar software-entiteiten, zoals agenten, informatie op een eenvoudige manier kunnen terugvinden, delen en integreren.

Resource Description Framework (RDF) is ontwikkeld door een subgroep van het W3C en is een markeertaal die in staat is informatie te beschrijven en te linken op het World Wide Web. RDF-documenten worden geschreven in XML. Een RDF-expressie is opgebouwd uit een 'Resource', een 'Property' en een 'Property Value' ook wel gekend als respectivelijk 'subject', 'predicate' en 'object'. Een vereenvoudigd voorbeeld zorgt ervoor dat het concept achter RDF duidelijk wordt. Onderstaand (vereenvoudigd) RDF-statement stelt software-entiteiten in staat om af te leiden dat het subject de fictieve pagina http://www.uhasselt.be/Agenttechnologie is en dat de auteur ervan Mario Aerts is. (W3CSchools, 2007; Palmer, 2001) Via RDF geeft men dus via 'predicaten' semantische informatie over bepaalde 'objecten'.

```
<?xml version="1.0"?>
<RDF>
  <Description about="http://www.uhasselt.be/Agenttechnologie">
    <author>Mario Aerts</author>
    </Description>
</RDF>
```

[XIV] http://setiathome.berkeley.edu/index.php

Een nieuwe standaard voor het Semantic Web die ook ontwikkeld wordt door een W3C-groep, namelijk de "Web Ontology Working Group", heet Web Ontology Language (OWL). Via OWL is het gemakkelijker om machines informatie te laten interpreteren omdat het extra woordenschat deelt bovenop de semantische regels zoals dat het geval is in RDF. OWL definieert dus een ontologie om formeel vast te leggen wat bepaalde termen in Web documenten betekenen. In dit opzicht is het een verbeterde versie van RDF en verwacht men ook dat OWL erg belangrijk zal zijn voor de ontwikkeling van het Semantic Web. (W3C, 2004)

HOOFDSTUK 6: JADE als agentplatform

In voorgaand hoofdstuk beschreven we de waarde die een agentplatform kan hebben voor het ontwikkelen van agenttoepassingen. In deze context wordt in dit hoofdstuk dieper ingegaan op het JADE-platform.

6.1 Beschrijving van het JADE-platform

JADE (Java Agent DEvelopment framework) is een bekend en reeds vaak gebruikt middleware-platform dat een Open Source framework is, uitgegeven onder de GNU Lesser licentie. JADE is volledig in Java ontwikkeld en opgebouwd naar de FIPA-richtlijnen. Het feit dat JADE voldoet aan de FIPA-standaarden zorgt ervoor dat applicaties die ontwikkeld worden op basis van JADE interoperabel zijn met andere applicaties die deze standaarden volgen. (JADE Website, 2007a) Zowel uit de literatuur als uit mijn interviews met de heer Vermeulen en dr. ir. Roos blijkt duidelijk dat zij JADE één van de gebruiksvriendelijkste en meest geschikte platformen vinden om agentapplicaties te ontwikkelen. Deze laatste bevinding samen met het voldoen aan de FIPA-standaarden zijn de belangrijkste reden waarom hier een bespreking van het JADE-platform gegeven wordt.

In Bellifemini et al. (2003) definieert Bellifemini, projectleider van het JADE framework, JADE als volgt: "JADE is an enabling technology, a middleware for the development and run-time execution of peer-to-peer applications which are based on the agents paradigm and which can seamless work and interoperate both in wired and wireless environment". In deze definitie komen enkele begrippen terug die we reeds besproken hebben. Hieronder vallen concepten als peer-to-peer applicaties, middleware, CORBA en Java RMI.

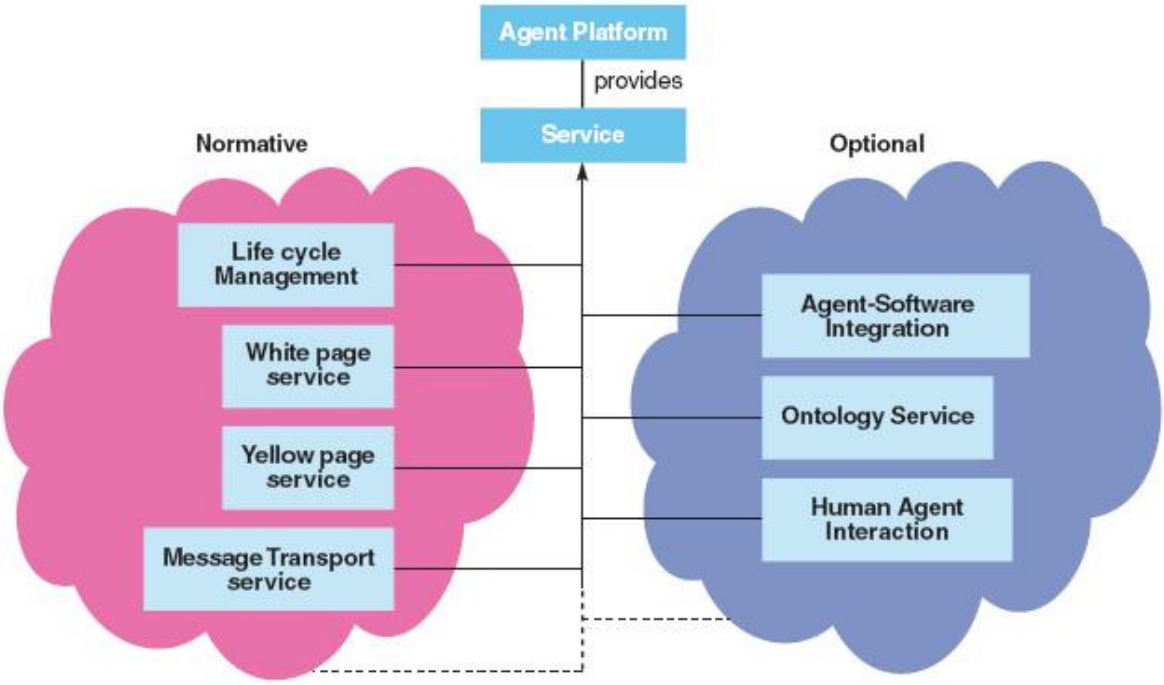
Het JADE-framework biedt functionaliteit om een dynamisch en gedistribueerd multi-agentsysteem te ontwikkelen. Het opzet van het JADE-team is de mogelijkheid bieden aan ontwikkelaars om relatief snel te komen tot een werkende en zinvolle bedrijfsapplicatie op basis van agenttechnologie. Via JADE kan men een belangrijke stap zetten van abstract onderzoek betreffende agenttechnologie in de richting van zinvolle en nuttige applicaties op basis van agenttechnologie. Op de site van JADE is men duidelijk over de manier waarop men dit wil bewerkstellingen: "The goal of JADE is to simplify the development of multi-agent systems while ensuring standard compliance through a comprehensive set of system services and agents in compliance with the FIPA specifications…". (JADE Website, 2007a)

In het resterende deel van dit hoofdstuk zullen we aantonen op welke manier het JADE-platform nuttig kan zijn om een eigen agentapplicatie te ontwikkelen. Vermits JADE voldoet aan de FIPA-standaarden, zal de structuur van JADE samenvallen zoals de FIPA-specificaties in hoofdstuk 4. We kunnen stellen dat JADE de FIPA-standaarden concretiseert in een Java bibliotheek.

6.1.1 Functionaliteit op systeemniveau die voorzien wordt door JADE

In onderstaande afbeelding vinden we met een beetje inlevingsvermogen de drie diensten[XV] terug die we beschreven hebben bij de 'FIPA Abstract Architecture Specification' onder de normatieve componenten van het JADE platform. We merken op dat een component voor het beheren van agenten en hun levenscyclus ook verplicht is bij JADE. We kunnen hierover zeggen dat FIPA vond dat dit proces moeilijk abstract te omschrijven is zodat het niet in de 'FIPA Abstract Architecture Specification' [00001] vermeld wordt. (FIPA Website, 2007a)

[XV] Deze drie diensten werden gedefinieerd als de agent-directory-service, message-transport-service en service-directory-service.



Figuur 20: Diensten op het JADE platform (Bellifemine et al., 2003)

De diensten voor de levenscyclus van agenten en de 'white-page'-dienst wordt door JADE ingericht volgens de FIPA Agent Management Specificatie. Concreet gezien komt het er op neer dat JADE twee agenten voorziet die Agent Management System (AMS) en Directory Facilitator (DF) genoemd worden. Over de Message Transport Service wordt in een volgende onderdeel meer uitleg gegeven. (JADE Website, 2007a)

Buiten voorgaande FIPA-specificaties definieert JADE ook nog andere tools die voorhanden zijn op het JADE AP. Zo voorziet het AP een Graphical User Interface (GUI) voor het management, opvolgen en controle van de status van de agent. Op deze manier kunnen gebruikers ingrijpen in de levenscyclus van agenten. Deze GUI is geïmplementeerd als een agent die de Remote Monitoring Agent (RMA) genoemd wordt. (JADE Website, 2007; Poggi et al., 2002)

Verder bezit het JADE-platform enkele grafische tools die kunnen gebruikt worden om te debuggen en belangrijke informatie over het AP te weten te komen. Naast de reeds aangehaalde tools, namelijk de RMA, AMS en DF bezit JADE volgende tools:

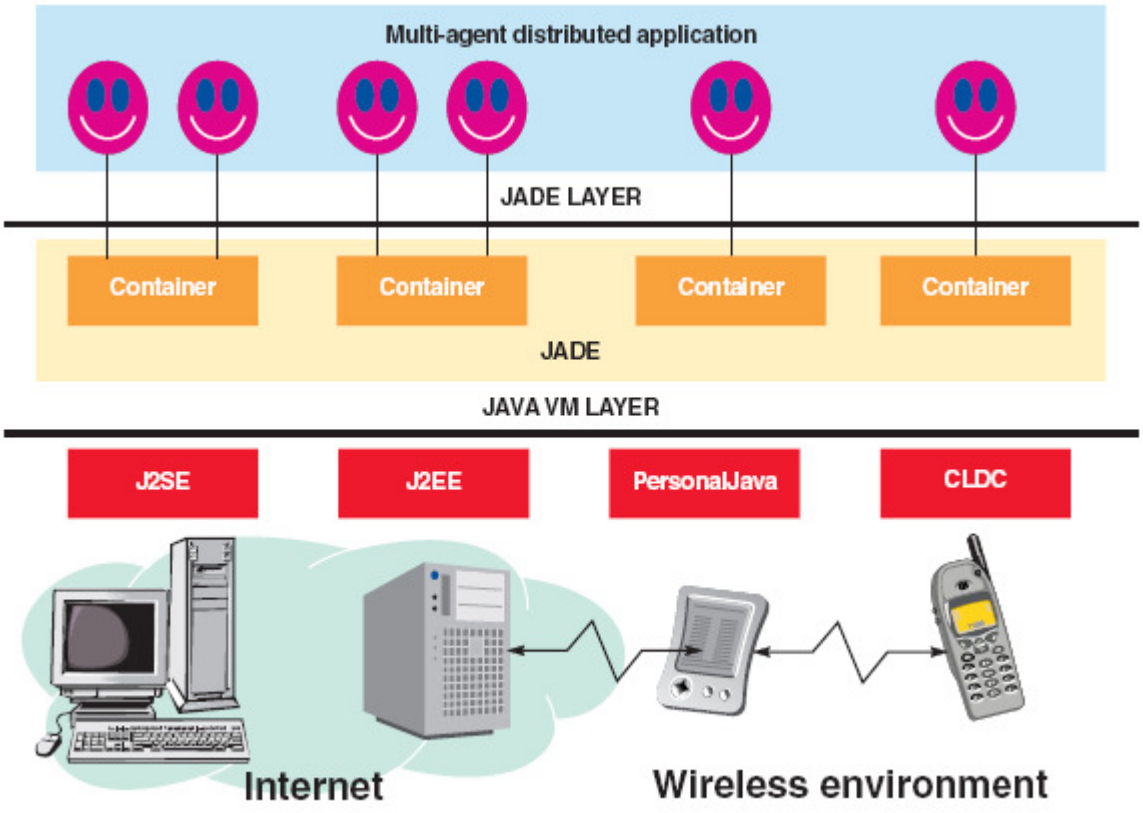
- ***Sniffer Agent**, a tool to sniff message exchange between agents. This tool is useful to debug a conversation between agents. It allows also to save the conversation to a file and load from a file.*
- ***Introspector Agent**, a tool to debug a specific agent. This tool allows to introspect the message queues (incoming and outgoing messages) and the agent tasks (the behavior according to the JADE abstraction). The tool allows also to control the execution of the agent, in particular stopping and executing slowly or step-by-step.*
- ***Dummy Agent**, a tool to compose custom messages to send and display the received messages. Messages can also be saved to a file and loaded from a file. The tool allows to simulate by hand the communicative behavior of an agent.*

(JADE Website, 2007a)

6.1.2 JADE's ondersteuning voor uitvoeren van multi-agentsystemen

Op technisch gebied is JADE volledig uitvoerbaar op een grote diversiteit aan systemen. JADE is ontwikkeld in Java en doet dus voor de uitvoering beroep op het Java-platform. JADE kan uitgevoerd worden op eender welk Javaplatform dus zowel op J2SE, als op J2EE en J2ME. Elk van deze platformen bezit een Java Virtual Machine (JVM). Dit is de omgeving waarin de Java-code uitgevoerd wordt. Deze JVM moet op elk toestel actief zijn en wordt door het JADE-team een container genoemd, wat overigens een logische naam is vermits zo een toestel echt agenten bevat ('contains'). Deze containers vormen samen het JADE-platform en verbergen voor zowel de actieve agenten als de ontwikkelaar de complexiteit in de verschillende lagen die actief zijn bij het draaien van het platform zoals hardware, netwerktechnologie, besturingssystemen,… (Bellifemine, 2003)

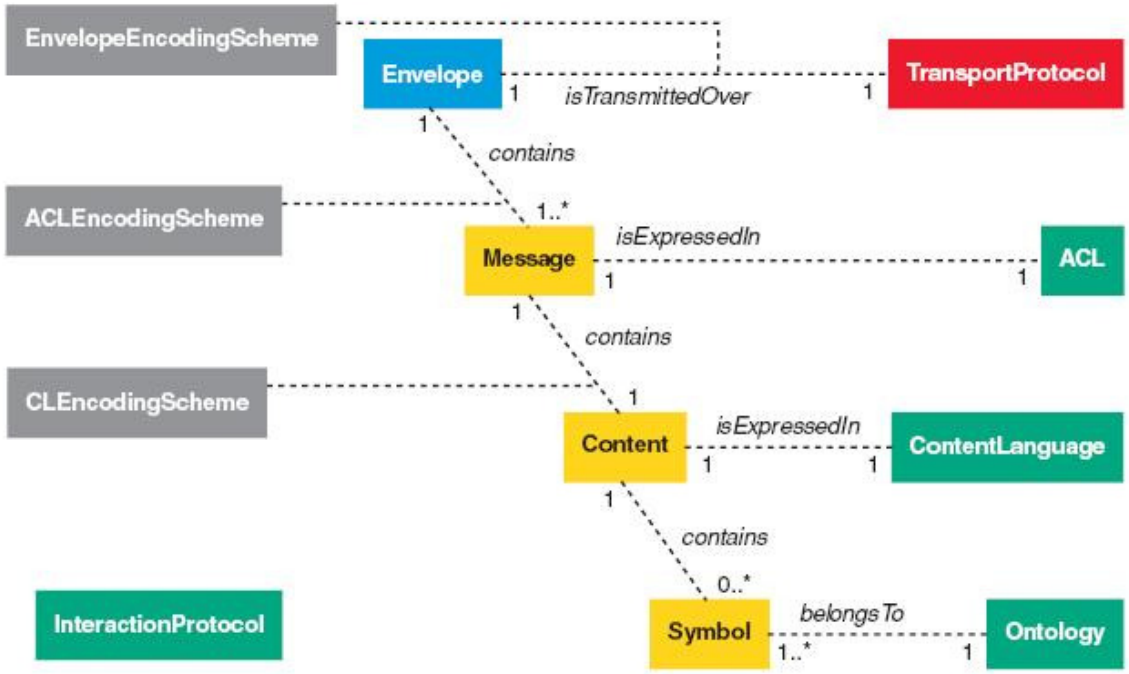
Deze opzet zorgt voor een grote compatibiliteit in zowel netwerkinfrastructuur als hardware-infrastructuur waarop JADE werkt. Doordat JADE uitvoerbaar is op verschillende Java-platformen wordt het mogelijk om verschillende hardwaretoestellen zoals servers, desktops en mobiele toestellen te integreren in één enkel systeem.



Figuur 21: Voorstelling van het JADE framework (Bellifemine et al., 2003)

6.1.3 Communicatie binnen het JADE platform

In onderstaande figuur 22 is het FIPA-communicatiemodel terug te vinden zoals het geïmplementeerd is in JADE. Het definieert de componenten die nodig zijn zodat agenten zinvol kunnen communiceren. We zullen deze figuur beknopt bespreken. *Interactieprotocollen* zijn formele uitwisselingsschema's van berichten die agenten dienen te volgen in een conversatie. Een ACL-bericht is uitgedrukt in een ACL en wordt gecodeerd tot een *'payload'* volgens een bepaalde *codeertaal*. De inhoud van een bericht moet volgens een bepaalde syntax opgebouwd zijn. Deze regels vinden we terug in een *inhoudstaal*. Verder dient er een woordenschat gedefinieerd te worden voor bepaalde domeinen zodat deze in berichten verwerkt kunnen worden. Deze link tussen realiteit en de berichten wordt gerealiseerd in een *ontologie*. De 'payload' wordt omkapseld door een *enveloppe* waarna het geheel via een *transportprotocol* verzonden kan worden.



Figuur 22: JADE implementatie van FIPA-berichtuitwisseling (Bellifemine et al., 2003)

Wanneer agenten op eenzelfde platform hun thuisbasis hebben, verloopt de communicatie niet noodzakelijk via FIPA-specificaties. Deze communicatie kan immers veel eenvoudiger plaatsvinden dan via de voorgeschreven FIPA-standaarden via Java objecten. Indien agenten op dezelfde container berichten willen uitwisselen kan deze communicatie immers via eenvoudige 'event-notification' gebeuren. Indien agenten echter op verschillende containers actief zijn, gebeurt de communicatie via het uitwisselen van Java-objecten die de berichtinhoud inkapselen. JADE maakt in dat geval gebruik van Java RMI voor berichtuitwisseling tussen agenten binnen hetzelfde platform maar op verschillende containers. (JADE Website, 2007b)

Communicatie tussen agentplatformen daarentegen verloopt via de FIPA-specificaties. Men valt dan terug op de CORBA-technologie om RPCs[XVI] uit te wisselen. Communicatie verloopt dus via het IIOP protocol tussen twee IDL 'stubs' van de platformen. Deze werkwijze houdt in dat het Java ACL-bericht object omgezet wordt naar een string waarna deze via IIOP de 'remote invocation' op de IDL 'stub' van het ontvangende platform kan doen. Het ontvangende platform haalt de methodeaanroep uit het IIOP pakket waarna de string terug 'geparsed' kan worden naar een ACL-bericht object in

[XVI] Remote procedure call

Java. Het platform kan nu dit Java object via een 'event' of een RMI-aanroep afleveren bij de geadresseerde agent. (JADE Website, 2007b) Communicatie tussen platformen wordt verzorgd door een JADE Agent Communication Channel (ACC) dat op elk platform terug te vinden is. De functionaliteiten die geleverd worden door de ACC zijn de volgende (Poggi et al. 2002):

- Integratie van Message Transport Protocols (MTPs)
- Routeren van berichten die binnenkomen en uitgaan
- Onafhankelijk van het gebruikte protocol berichten verwerken

Omdat transportprotocollen via Java-klassen toegevoegd kunnen worden aan een AP, kan een AP via meerdere adressen bereikt worden. Afhankelijk van het formaat van het bericht zal het binnenkomen via een andere poort van het AP. Het is evident dat dit complex geheel een routeerdienst nodig heeft. Wanneer een bericht een AP bereikt door een poort, zal de ACC aan de hand van de Agent ID het bericht doorsturen naar de geschikte agent container waar de geadresseerde agent zich bevindt. Op die manier zorgt de ACC voor een routeerservice die ten minste één verbinding verder draagt. Door het agent ID te controleren kan een ACC afleiden naar welk AP een bericht gestuurd kan worden. Via dit AID is het ook mogelijk om het MTP vast te stellen dat gebruikt dient te worden. Dit kan eenvoudig gebeuren omdat FIPA-adressen opgebouwd zijn via URLs waarlangs een ACC het MTP kan achterhalen. (JADE Website, 2007b; Poggi et al., 2002)

6.2 JADE API

JADE voorziet een API (Application Progamming Interface) die door de mogelijkheden van Java gebruikt kan worden om snel en relatief eenvoudig een multi-agentsysteem te maken. De JADE API is terug te vinden op de JADE website[XVII] maar kan zelf ook gegenereerd worden door het importeren van de JADE bibliotheken in een Java-ontwikkelomgeving. Verder werd er beroep gedaan op de JADE PROGRAMMER'S GUIDE (JADE board, 2006a) en JADE ADMINISTRATOR'S GUIDE (JADE board, 2006b) voor extra verrijkende informatie over de JADE API. In dit deel van dit hoofdstuk zullen we bespreken op welke manier de JADE-klassen de verwachtingen invullen die we hebben na

[XVII] http://jade.tilab.com/doc/api/index.html

de bespreking van de rol van agent-middleware en de functionaliteit van het JADE platform. Dit deel veronderstelt dat de lezer een basiskennis Java heeft.

Een API definieert enerzijds klassen waarvan de functionaliteit dadelijk in een multi-agentsysteem gebruikt kan worden en anderzijds abstracte klassen en interfaces die de programmeur moet implementeren en zelf coderen. De JADE API bestaat uit volgende 'packages' (verzameling van logisch samenhangende klassen in één "pakket"):

- `jade.core` : kernel van het systeem met fundamentele klassen zoals die van een `Agent` en een `AID` (Agent IDentifier)
    - `jade.core.behaviours` : klassen die gebruikt kunnen worden om agenten een gedrag te geven. De `behaviours` waarmee gewerkt kan worden, geven de agenten taken en intenties. We kunnen dus stellen dat we via deze klassen agenten een vorm van BDI-structuur kunnen geven
- `jade.lang`
    - `jade.lang.acl` : klassen die de concepten van de FIPA-ACL implementeren
- `jade.content` : klassen die ondersteuning geven aan ontologieën en content-languages
    - `jade.content.lang.sl` : bevat de codec voor de SL content language. Met een codec kan een bericht geprint worden (geparsed) naar de SL-taal maar ook een bericht in de SL-taal ontcijferd worden (encoded) naar een Java-object
- `jade.domain` : definieert klassen voor de agenten die systeemtaken uitvoeren volgens de FIPA-standaarden, namelijk de AMS- en DF-agent
    - `jade.domain.FIPAAgentManagement` : bezit klassen die het concept van FIPA Agent Management voorstellen door constanten te definiëren uit de `FIPA-Agent-Management` ontologie.
    - `jade.domain.JADEAgentManagement` : deze 'package' bezit de klassen die de concepten van de JADE ontologie voor agentmanagement voorstellen. Deze klassen stellen de acties in het JADE-platform voor met de daaraan gekoppelde ontologie. Zo zijn er bijvoorbeeld de klassen `ShowGui` (DF toont zijn GUI), `CreateAgent` (er wordt een nieuwe agent gemaakt op het

platform), `InstallMTP` (AMS installeert nieuwe MTP voor inter-platform communicatie)

- `jade.gui` : definieert GUI's van alle slag voor het JADE platform.
- `jade.mtp` : via deze klassen kan men MTPs integreren in het JADE-framework. Ook de implementatie van deze protocols vinden we hier terug.
- `jade.proto` : klassen die het gebruik van interactieprotocols ondersteunt
- `jade.wrapper` : omvat klasses die gebruikt kunnen worden door externe applicaties die de JADE-functionaliteit gebruiken
- `jade.tools` : de tools voor debuggen en het bijhouden van de administratie van het AP die we besproken hebben in voorgaand hoofdstuk kunnen via deze klassen geïmplementeerd worden

6.2.1 Agent management in JADE

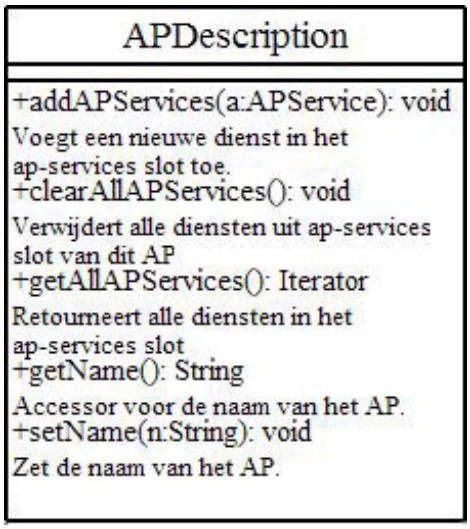
De klassen in de 'package' `jade.domain.FIPAAgentManagement` modelleren de FIPA specificaties betreffende agentmanagement. In deze 'package' vinden we klassen terug die gebruikt worden om de componenten en acties in een AP voor te stellen zoals deze beschreven zijn in de `FIPA-Agent-Management` ontologie. In deze ontologie worden, zoals we gezien hebben, niet alleen componenten maar ook acties en 'exceptions' gedefinieerd en met kenmerken beschreven.

De componenten worden in JADE samen met hun parameters en slots opgenomen in een woordenlijst met constanten die gekoppeld zijn aan stringwaarden. Deze stringwaarden komen overeen met de benamingen van de objecten, acties en 'exceptions' die gedefinieerd worden in de FIPA-specificaties. Deze woordenlijst is gedefinieerd in de interface `FIPAManagementVocabulary` die geïmplementeerd wordt in de klasse `FIPAManagementOntology`. Naast het implementeren van deze interface breidt de `FIPAManagementOntology` ook nog de `Ontology` klasse uit. Instanties van deze klasse omkapselen de velden en methoden van een ontologie in JADE.

Naast een ontologie om te communiceren met deze componenten is er natuurlijk ook een inhoudstaal en een voorgeschreven 'communicative act' nodig. FIPA definieerde hiervoor de `FIPA-SL0` inhoudstaal en het `fipa-request` interactieprotocol. In JADE worden deze

specificaties geconcretiseerd door instanties aan te maken van de klasse `jade.content.lang.sl.SLCodec` en de klassen die instaan voor het implementeren van de intitiator- en antwoordrol van het `fipa-request` protocol in het pakket `jade.proto`.

In de twee bovenstaande alinea's is uit de doeken gedaan op welke manier agenten interactie kunnen hebben met systeemcomponenten. Een voorbeeld van een implementatie van een component is de klasse `APDescription`. Instanties van deze klasse bevatten de data en gegevensstructuren die eigen zijn aan een AP. Een voorstelling in Unified Modelling Language (UML)[XVIII] van deze klasse heb ik ingevoegd onder figuur 23. We vinden er ook de klassen terug waar de ontologie en concepten van andere componenten vastgelegd worden. In de bespreking van de FIPA-standaarden hebben we de concrete ontologie en beschrijving van enkele componenten getoond. Andere klassen die componenten uit de agentmanagement-specificatie van FIPA implementeren zijn de klassen `AMSAgentDescription`, `APService`, `Deregister` en `DFAgentDescription`[XIX]. Twee belangrijke componenten, namelijk de AMS- en DF-agent zullen we in volgende paragraaf nader bespreken.



Figuur 23: UML van de klasse die het AP beschrijft

[XVIII] Enkel de publieke velden en methoden zijn weergegeven zoals ze terug te vinden zijn in de Javadocs. Deze opmerking geldt voor alle UML-diagrammen in dit hoofdstuk.

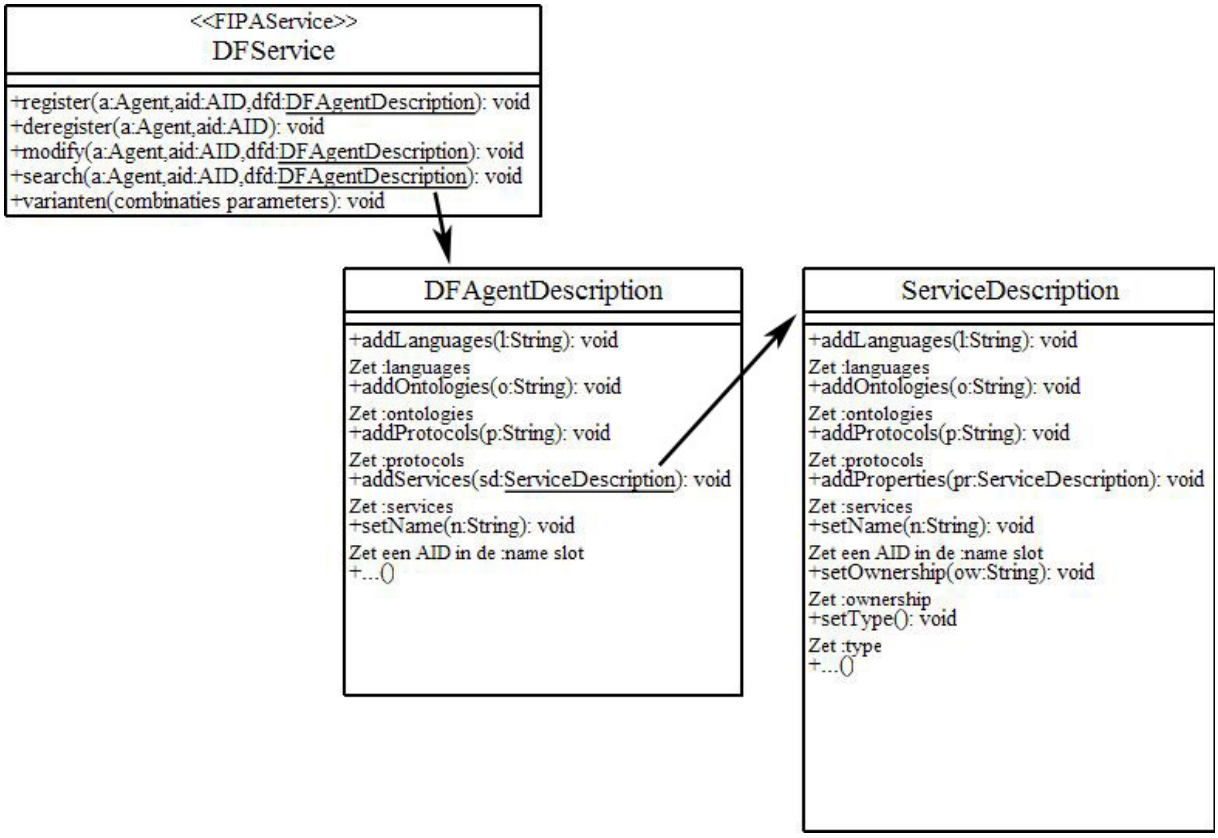
[XIX] Enkel de omschrijving van de AID wordt niet als klasse geïmplementeerd in de package `jade.domain.FIPAAgentManagement`. De implementatie van dit element uit de specificatie vinden we immers terug in het package jade.core.AID.

DF- en AMS-Service

Zoals we reeds aangehaald hebben, wordt de 'yellow-pages'-dienst bij JADE geïmplementeerd via een DF-agent. Deze agent op systeemniveau zorgt ervoor dat agenten hun diensten kunnen registreren zodat deze opvraagbaar en gebruikt kunnen worden door andere agenten. Het beschrijven van een dienst zelf gebeurt via de `ServiceDescription` klasse. Deze klasse waarin het type, de naam en allerlei andere regels gedefinieerd worden, is opgebouwd naar model van het `service-description` object in de `FIPA-Agent-Management` ontologie.

Agenten die hun diensten bij de DF-agent registeren, gebruiken hiervoor de specificaties in de `df-agentdescription` vermeld in de `FIPA-Agent-Management` ontologie. De klasse `jade.domain.FIPAAgentManagement.DFAgentDescription` modelleert de nodige velden en concepten van de ontologie om de geregistreerde agenten en hun diensten te omschrijven. Deze klasse bevat methoden om de slots uit de FIPA-specificaties op te vragen of in te vullen. Een instantie van deze klasse is dus een element van een DF-agent die één of meerdere geregistreerde diensten van een bepaalde agent bij deze DF-agent omvat.

De `jade.domain.DFService` klasse definieert methoden om de diensten van agenten bij de DF-agent te registreren, aan te passen, te verwijderen of op te zoeken. Deze methoden worden respectievelijk `register()`, `modify()`, `deregister()` en `search()`. Voor elke methode zijn meerdere exemplaren beschikbaar daar combinaties van meerdere parameters beschikbaar zijn. Logischerwijs zijn vaak terugkomende parameters instanties van de klasse `Agent`, `AID` of `DFAgentDescription`. Een overzicht van deze klassen die de werking van de DF-agent implementeren, is terug te vinden in figuur 24. De klassen zijn in een UML-voorstelling weergegeven waarin enkel de meest relevante methoden zijn opgenomen.



Figuur 24: JADE klassen die betrokken zijn in de werking van de DF-agent

We hebben gezien dat de AMS als een soort administrator werkt op het AP dat FIPA voor ogen heeft. Hierbij levert hij de zogenoemde 'white-page'-diensten. Net zoals de DF-agent zijn instanties van de klasse `jade.domain.AMSService` de AMS-agent op een AP. Verder worden de geregistreerde agenten opgeslaan in instanties van de klasse `jade.domain.FIPAAgentManagement.AMSAgentDescription`. We zien in deze beschrijving duidelijk de gelijkaardige aanpak van de interactie die agenten dienen te hebben met een DF- en AMS-agent.

6.2.2 Aanmaken van een agent in JADE

Een agent kan een programmeur makkelijk aanmaken door gebruik te maken van `jade.core.Agent` als basisklasse. Op deze manier kan men mogelijkheden overerven om basisinteracties op te zetten met het agentsysteem waar de agent zich op bevindt (registratie, configuratie, remote management, …). Dit wordt mede mogelijk gemaakt door de makkelijkheid waarmee de AMS en DF-agent opgevraagd kunnen worden via

`getDefaultDF():AID` en `getAMS():AID`. Verder worden door deze overerving methodes bekomen om het specifieke gedrag van een agent te programmeren. We merken ook op dat er methodes om de levenscyclus van een agent te veranderen in de API van de `Agent` klasse aanwezig zijn. Deze methodes hebben de vorm `doxxx()` met op de kruisjes een status te kiezen uit INITIATED, ACTIVE, SUSPENDED, WAITING, DELETED, TRANSIT.

Om een agent te kunnen identificeren voorziet JADE bij het aanmaken van een agent een klasse AID (`jade.core.AID`). Vermits dit identificatiemiddel belangrijk is voor het uitwisselen van berichten, wordt deze opgebouwd in twee delen, namelijk de lokale naam en de platformnaam zodat we de syntax `<local-name>@<platform-name>` krijgen. Via dit identificatiemiddel kan de agent overigens geregistreerd worden bij de AMS. Vervolgens wordt de `setup()` methode uitgevoerd waarmee de eerste taken toegewezen kunnen worden aan de agent. Zo wordt de agent geregistreerd bij de AMS, wordt de status op actief gezet en dient men verplicht een `Behaviour` te definiëren. Via andere belangrijke methodes in de `jade.core.Agent` kan men argumenten meegeven aan een agent, de werking van een agent beëindigen en de agent taken laten uitvoeren. Dit laatste gebeurt aan de hand van de methode `addBehaviour()`.

Een ACL bericht 'msg' kan door een agent verstuurd worden door de `Agent.send(ACLMessage msg)` methode aan te roepen. De Java klasse `jade.lang.acl.ACLMessage` vormt de uit te wisselen 'payload'. In deze klasse zijn statische variabelen gedefinieerd die overeenkomen met de 'communicative acts' zoals die gedefinieerd zijn door FIPA. De constructor van de `ACLMessage` klasse vereist dat er bij het aanmaken van een agent een 'communicative act' opgegeven wordt. Verder zijn er accessormethodes en mutatormethodes gedefinieerd voor de 'slots' van een ACL-bericht zoals die gedefinieerd zijn in de specificatie voor de FIPA-ACL. Verder zijn FIPA-specificaties op het gebied van reply-berichten erg streng. JADE heeft dit opgelost door een `createReply()` methode te definieren in de `ACLMessage` klasse.

Om een goed begrip te verzekeren wil ik nog even de nadruk leggen op de manier waarop agenten binnen verschillende platformen nu interoperabel kunnen zijn. Via JADE maken we immers een Java-object aan dat overeenkomt met de FIPA-specificaties voor ACL berichten. Het is logisch dat dit Java-object niet verstuurd wordt. Dit zou immers repercussies hebben voor de interoperabiliteit met andere platformen. Daarom wordt

door het AP de `jade.lang.acl.StringACLCodec` aangeroepen om het bericht in een string te om te zetten (parsen). Op deze manier kunnen alle agenten op verschillende APs deze berichten consistent interpreteren. Deze klasse kan ook gebruikt worden om een String om te zetten naar een Java-object.

Elke agent heeft zijn eigen datastructuur waar zijn persoonlijke ACL-berichten in terug te vinden zijn. Door de `receive():ACLMessage` methode van een agent te definiëren in een actief `Behaviour` kan deze zijn lijst van berichten verwerken. Deze lijst van berichten kan ook geadresseerd worden op een manier waarop alle andere processen van de agent geblokt worden, namelijk via `blockingReceive():ACLMessage`. Agenten die versturen en verwerken van berichten als taak hebben, kunnen de voorgedefinieerde `behaviours ReceiverBehaviour` of `SenderBehaviour` implementeren.

De manier waarop de lijst van berichten die eigen zijn aan een agent verwerkt wordt, verloopt via het principe van First-In-First-Out (FIFO). Toch is dit niet altijd wenselijk en is het wenselijk dat men bepaalde berichten eerst gaat verwerken. Deze mogelijkheid wordt gegeven door de klasse `MessageTemplate`. In deze klasse zijn methodes gedefinieerd om de 'slots' van de inkomende ACL-berichten te vergelijken met een voorgedefinieerde waarde. Deze methoden zijn van de vorm `Matchxxx()` waarbij de kruisjes worden ingevuld door het label van het slot in een ACL-bericht in de FIPA-ACL specificatie. Verder is het ook mogelijk om een specifieke patroon te definiëren waarop inkomende berichten doorzocht worden. Dit is mogelijk door een klasse te ontwerpen die de interface `MessageTemplate.MatchExpression` implementeert. Op deze manier kan bijvoorbeeld onderzocht worden of een bericht al dan niet tot een bepaald interactieprotocol behoort door een template te maken waar het betreffende protocol in het `:protocol` slot opgegeven is.

6.2.3 Toewijzen van gedrag aan een agent

Per agent voorziet JADE een Java-thread. In deze thread kan een agent meerdere taken of activiteiten uitvoeren die men kan implementeren als `Behaviour` objecten. Het op elkaar afstemmen van deze taken wordt geregeld door de `Agent` klasse die een soort van planningsmodule heeft die afgeschermd is van de programmeur. We kunnen nog

meedelen dat de procedure waarmee deze planner zorgt voor de uitvoering van de `Behaviours` een willekeurig proces is (Round-Robin). Door deze `Behaviours` de geschikte toegangsrechten te geven, wordt de autonomie aan de agent verleent. Hij heeft immers alleen toegang tot zijn eigen gedragingen en controleert zelf de uitvoering van zijn taken zonder dat andere 'objecten' hierop directe invloed kunnen uitoefenen.

Het klassepakket `jade.core.behaviours` voorziet allerlei klassen waarmee we gedragingen kunnen definiëren voor een agent. Een agent moet verschillende taken kunnen uitvoeren. Doordat deze processen in dezelfde Java thread uitgevoerd worden, kan een uitvoering niet pre-ëmptief zijn en dient de `action()` methode van een `Behaviour` steeds afgelopen te zijn alvorens over te schakelen naar een ander `Behaviour`. Nadat de `action()` methode klaar is wordt de `done()` methode aangeroepen om te controleren of de taak volbracht is. Is dit niet het geval plaatst de planner deze taak terug in de lijst met de nog te voltooien taken en is de status van het gedrag geblokkeerd.

Er is een abstracte superklasse, namelijk `Behaviour`, gedefinieerd die alle mogelijkheden open laat om zelf een subklasse te ontwerpen. Programmeurs erven dan de methode `action()` over en schrijven hier code voor. Wanneer een bepaald gedrag wacht op een bepaald bericht kan het zijn status via de methode `block()` blokkeren. De planner zet het gedrag/taak dan in de lijst met de geblokte taken. Alle taken in deze lijst worden terug geactiveerd als er een bepaalde gebeurtenis voorvalt. Deze gebeurtenissen zijn op dit moment gedefinieerd als één van volgende activiteiten zich voordoet, namelijk het binnenkomen van een ACL-bericht, een timeout die geprogrammeerd is voor een bepaalde taak die vervalt of het aanroepen van de `restart()` methode van een `Behaviour`. Verder heeft JADE een object voorzien om gegevens uit te wisselen tussen taken in de vorm van een 'datastore' die via de methoden `get/setDataStore()` gebruikt kan worden.

Aangezien een agent altijd de interne status van zijn eigen gedragingen dient bij te houden is het niet evident om complexe gedragingen te modelleren. We vinden dan ook klassen terug die functionaliteiten voorzien zodat specifieke gedragingen eenvoudiger geïmplementeerd kunnen worden. Zo is er bijvoorbeeld de klasse `CyclicBehaviour` die zelf reeds overerft van `SimpleBehaviour` en atomische taken definieert die in een altijd

durende lus worden uitgevoerd. De `done()` methode van deze klasse heeft dus steeds `false` als resultaat. Een andere soort van voorgedefinieerd gedrag is het `OneShotBehaviour`. Dit gedrag wordt dadelijk uitgevoerd en de methode `done()` is dus altijd `true`. Via de klasse `WakerBehaviour` kan men agenten geprogrammeerde acties eenmalig laten uitvoeren na een bepaalde periode via een speciale `onWake()` methode. Een andere voorgedefinieerde klasse is `TickerBehaviour` die ervoor zorgt dat een bepaalde actie continu na een bepaalde periode uitgevoerd wordt (`onTick()`).

Tot slot kunnen we zeggen dat elke `Behaviour` een variabele `myAgent` heeft die verwijst naar de agent die het gedrag uitvoert.

6.2.4 Interacties tussen agenten

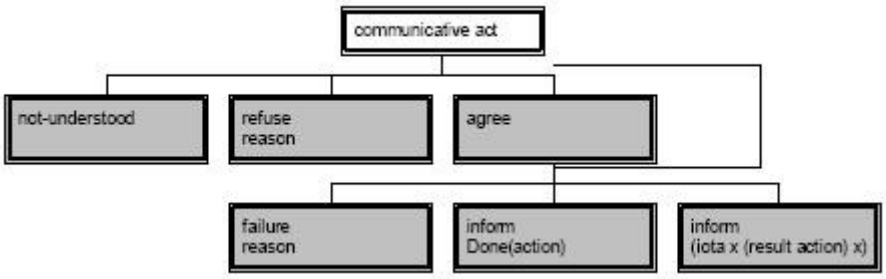
We hebben nu reeds gezien hoe het mogelijk is om via het JADE-platform berichten te versturen. Tijdens agentonderhandelingen of conversaties is het zo dat er vaak dezelfde patronen of interacties terugkomen. Verder moet het mogelijk zijn om berichten te plaatsen in een context van conversaties en onderhandelingen. Daarom zijn er in het JADE-platform bepaalde klassen voorzien die voorgedefinieerde interacties mogelijk maken. Aangezien een agent in de rol van initiator (Initiator) of antwoorder (Responder) kan ingeschakeld worden, voorziet JADE klassen voor elk van beide rollen. Deze klassen zijn terug te vinden in het `jade.proto` package. Men kan het protocol van een bericht definiëren in het `:protocol` slot door middel van de interface `jade.domain.FIPANames.InteractionProtocol`. De code om een bericht te versturen in de context van een FIPA-Request zou dan als volgt zijn: `ACLMessage.setProtocol(FIPANames.InteractionProtocol.FIPA_REQUEST).`

Ter illustratie zullen we beschrijven welke ondersteuning JADE kan bieden voor enkele protocollen, waaronder het FIPA-Contract-Net protocol.

JADE AchieveRE

JADE definieert een initiator- en responderklasse voor de protocols die zij omschrijven als AchieveRE (Rational Effect). Deze klassen bieden ondersteuning voor alle FIPA-Request

interactie protocols. Hieronder vallen de interactieprotocollen FIPA-Request, FIPA-query, FIPA-Request-When, FIPA-Recruiting en FIPA-brokering. De ondersteuning van al deze interactieprotocollen door slechts één initiator en respondentklasse is volgens het team van JADE mogelijk omdat deze protocollen een gelijkaardige structuur hebben. De kern van deze structuur is dat een initialiserende agent de uitwerking van zijn 'communicative act' op de ontvangende agent vergelijkt met zijn eigen intentie tijdens het versturen van dit bericht. In onderstaande figuur vinden we een voorstelling van deze structuur. Via dit interactieprotocol krijgt de agent een antwoord van de ontvanger van zijn bericht. Op die manier kan hij afleiden welk effect zijn bericht teweeggebracht heeft bij de andere agent. Dit noemt men ook wel het 'Rational Effect'.



Figuur 25: Structuur van interactieprotocollen ondersteund door AchieveRE klassen (JADE board, 2006a)

De klasse `AchieveREInitiator` bevat verder nog de methodes van de vorm `handlexxx(ACLMessage …)` met op de kruisjes het inkomende bericht. In deze procedure kan de programmeur de processen die dienen te gebeuren bij het ontvangen van een bepaald bericht definiëren. Zo wordt de methode `handleAgree(ACLMessage msg)` elke keer aangeroepen als er een `agree`-bericht ontvangen wordt. Verder is het mogelijk dat specifieke `Behaviours` gedefinieerd worden wanneer een bepaald bericht ontvangen wordt. Dit is mogelijk via de methode `registerHandlexxx(Behaviour b)`.

De klasse `AchieveREResponder` implementeert de Responder rol van de hierboven aangehaalde FIPA-protocollen. Bij het aanmaken van een nieuwe instantie van deze klasse dient men een toepasselijke `MessageTemplate` op te geven in de constructor. Deze klasse dient maar één afhandelingsprocedure te bezitten vermits alle berichten die deze Behaviour activeren het initialisatiebericht van het protocol is. Deze methode is gedefinieerd als `handleRequest()`. Verder is het weer mogelijk om nieuwe aangepaste Behaviours te definieren via `registerHandleRequest(Behaviour b)`.

FIPA-Contract-Net

De bespreking van de FIPA-specificatie van het Contract-Net protocol hebben we reeds gegeven. Dit protocol dient om een initiërende agent de mogelijkheid te geven bepaalde offertes te vergelijken van agenten die de gevraagde dienst hopen te kunnen leveren.

De initiator rol in dit protocol zit vervat in de klasse ContractNetInitiator. Bij het aanmaken van een object van deze klasse dient men het te versturen ACL-bericht mee te geven in de constructor, namelijk `ContractNetInitiator(Agent a, ACLMessage cfp)`. Zoals we gezien hebben is het eerste bericht een Call for Proposal die aan één of meerdere agenten gestuurd kan worden.

Aangezien een bericht dus aan meerdere agenten verstuurd moet kunnen worden, wordt in de methode `prepareCfs(ACLMessage cfp)` een `Vector` geretourneerd waarin deze berichten klaargemaakt worden om te verzenden.

Verder zijn er, net zoals bij de reeds besproken AchieveREInitiator klasse, een aantal 'callback' methoden gedefinieerd die aangeroepen worden wanneer een bepaald type van bericht ontvangen wordt. Zo wordt de methode `handlePropose()` aangeroepen wanneer er een propose bericht ontvangen wordt. We herinneren ons dat het dan mogelijk is om deze methode te overschrijven in onze eigen gedefinieerde klasse. Verder zijn er weer methodes gedefinieerd om een specifiek `Behaviour` te registreren om een bepaalde procedure in gang te zetten bij het ontvangen van een bericht.

Bij de klasse die de rol van 'Responder' op zich neemt, is het belangrijk een goed `MessageTemplate` op te maken om mee te geven aan de constructor zodat enkel de juiste berichten verwerkt worden. Om in te spelen op elke mogelijke toestand die een Responder kan hebben in dit protocol, kunnen de methodes `preparexxx()` geïmplementeerd worden. Op deze manier kunnen ACL-berichten verstuurd worden. Verder is ook weer de handige methode `createReply()` gedefinieerd.

6.2.5 JADE berichtenuitwisseling

We hebben de ondersteuning die JADE biedt om ACL-berichten volgens FIPA specificaties op te bouwen reeds besproken. Rest ons nog de belangrijke topics betreffende het transport en de codering van de berichten te bespreken.

Het gebruiken van transportprotocollen voor inter-platform berichtuitwisseling is een complexe aangelegenheid. Toch kan men flexibel en eenvoudig MTPs inpluggen op het JADE-platform. Het is zo dat een ontwikkeld MTP dat aan de FIPA-specificaties voldoet afzonderlijk gecompileerd en als een JAR-file gebruikt kan worden. Na de activatie van deze MTP is dit platform via een nieuw adres toegankelijk. Dit adres wordt bijgevoegd in het profiel van het platform (opvraagbaar bij de AMS via `get-description`) en toegevoegd aan alle systeemagenten (gedefinieerd via de `ams-agent-description`) die beheerd worden door de AMS. De activatie van een MTP kan gebeuren bij de installatie van een container op een machine via de command-line ofwel via de JADE RMA. Bij deze activatie dient men verplicht de Java-klasse die het protocol implementeert te specificeren. Optioneel kan men het transportadres meegeven van deze container. Als men dit niet doet, krijgt de container een standaardadres toegewezen. Bij het gebruik van de RMA is het natuurlijk nodig dat men de container waar men een protocol wil installeren specificeert. Volgende code stelt een container in staat om via IIOP berichten uit te wisselen en dit door middel van een ORBacus[XX]-gebaseerde IIOP MTP te implementeren op een vast adres:

```
-mtp
orbacus.MessageTransportProtocol(corbaloc:iiop:sharon.cselt.it:1234/j
ade)
```

Deze implementatie van ORBacus is beschikbaar als add-on op het JADE-platform. In de package `orbacus` zit de klasse `MessageTransportProtocol` die een API vormt voor ORBacus ORB implementatie. In bovenstaand voorbeeld is één van de drie mogelijke voorstellingen opgenomen om het IIOP-adres mee te geven in de constructor van de `MessageTransportProtocol`. De syntax van deze OMG-specificatie voor zo een transportadres is “`corbaloc: “ HostName “: “ portNumber “/” objectID`. Verder ondersteunt deze klasse `IOR:` en `corbaname:` adressen die terug te vinden zijn in de OMG-specificaties.

[XX] Versie van CORBA 2.6 implementatie.

Basis IIOP MTP

De basis IIOP MTP komt bij JADE als een aparte JAR-file. De MTP implementatie van deze Corba ORB is vervat in `jade.mtp.iiop.MessageTransportProtocol`. We merken dat deze klasse beperkt is in die zin dat het onmogelijk is een transportadres te specificeren bij de installatie.[XXI]

Andere MTPs

Een add-on die een MTP implementeert volgens de FIPA-specificatie voor het HTTP protocol, is downloadbaar op de JADE Web Site. Meer informatie over het gebruik en de installatie van deze module om ook berichten te kunnen sturen via HTTP is terug te vinden op de JADE Website.

Verder voorziet JADE ook interfaces in de 'package' `jade.mtp` die gebruikt kunnen worden om zelf MTPs te ontwikkelen en via een JAR-file als plugin te gebruiken. De MTP interface geeft de mogelijkheid een kanaal op te richten om ACL-berichten te verzenden en te ontvangen. Verder is er de `TransportAddress` interface die een representatie van een transportadres voorstelt. Op deze manier worden de reeds besproken onderdelen van een transportadres omkapseld in een object. Op die manier kan men het protocol, host, poort en object ook apart opvragen.

6.2.6 ACL berichtcodering

Standaard worden berichten in JADE gecodeerd in het stringformaat dat gedefinieerd is door FIPA. Toch is het mogelijk additionele klassen toe te voegen die ACL-berichten in een ander formaat coderen. Deze objecten worden ook wel ACLCodecs genoemd. De codering van een bericht dient meegegeven te worden in het :`acl-Representation` veld van de enveloppe zoals gezien in figuur 10.

[XXI] Een vergetelheid vanwege JADE zorgde ervoor dat het package jade.mtp niet terug te vinden is in de online-documentatie van de JADE v.3.4.1

Populaire codecs die van de JADE-site als add-on afgehaald kunnen worden zijn de XML Codec en de Bit Efficiënt ACL Codec. Naast constanten om de slots en waardes van een ACL-bericht voor te stellen, voorzien dergelijke codecs methodes in de aard van `encode()` en `decode()` om berichten te kunnen omzetten van de FIPA-specificatie naar een JADE ACLMessage object. Zo zet de methode `encode()` in de XML codec een ACLMessage object om naar het XML-formaat, klaar om te versturen in de vorm van bytes.

6.3 Bespreking JADE als agentplatform

De bibliotheekklassen in JADE zijn erg geschikt om multi-agentsystemen te ontwikkelen. Via overerving of zelfs het volledig gebruik van de ter beschikking gestelde klassen, kunnen we een agenttoepassing maken die grotendeels voldoet aan de kenmerken van het agentparadigma zoals we dit beschreven hebben in voorgaande hoofdstukken. Via de manier van toewijzing van taken en gedrag aan agenten kunnen we immers zorgen voor autonomie, reactiviteit en pro-activiteit van agenten. Bij het opbouwen van een agentplatform zal JADE bovendien automatisch één thread per agent reserveren zodat asynchrone procesafhandeling mogelijk wordt. De gedistribueerde architectuur wordt dan weer gegarandeerd door de mogelijkheid om een agentplatform op te splitsen in meerdere containers die beheerd worden door JADE en waar agenten actief op kunnen zijn. Verder kunnen we ook op eenvoudige manier de rollen die agenten bekleden in interactieprotocollen implementeren via de klassen die de FIPA-specificaties concretiseren.

Zoals we gezien hebben is het volledige FIPA-communicatiemodel geïmplementeerd in JADE. Door deze klassen kunnen we relatief eenvoudig en intuïtief ACL-berichten construeren en versturen. Hierbij worden complexe transportprocessen van het bericht en routeringsbeslissingen afgeschermd van de ontwikkelaar. Algemeen kunnen we stellen dat JADE erin geslaagd is om een platform te ontwikkelen zodat ontwikkelaars zich kunnen concentreren op de logica van de applicatie. Via JADE kan men een belangrijke stap zetten van abstract onderzoek betreffende agenttechnologie in de richting van zinvolle en nuttige applicaties op basis van agenttechnologie. Dit is mogelijk doordat de fundamentele basisconcepten van het agentparadigma automatisch worden overgeërfd in

de eigen applicatie en technologische overwegingen grotendeels ingevuld worden door JADE of met eenvoudige interfaces gedefinieerd zijn.

Aangezien JADE gebaseerd is op de FIPA-specificaties moeten we erkennen dat de ondersteuning voor het gebruik van ontologieën ook nog te beperkt is. Op het gebied van integratie van agenttechnologie en Web Services heeft JADE eind november 2006 een nieuwe add-on beschikbaar gesteld. Deze Web Service Integration Gateway (WSIG) is een poging om Web Services te integreren met agenten. Via deze module wordt het mogelijk om een agentdienst op de DF te registreren in het UDDI-register van een Web Service en omgekeerd. Verder kunnen beide technologieën gebruik maken van elkaars diensten. (Greenwood, 2005; Whitestein Technologies AG, 2005) Alhoewel FIPA pas in 2007 op de proppen wil komen met een specificatie die het gebruik van Web Services gecombineerd met agenten wil standaardiseren heeft JADE reeds in 2005 hiervoor een mechanisme geïmplementeerd. Welke de rol en de veranderingen in deze module zullen zijn ten gevolge van de FIPA-standaard die eraan zit te komen is nog niet zeker en zal opgevolgd moeten worden. Het is ook daarom dat we slechts oppervlakkig de WSIG-add-on besproken hebben.

Tot slot kwamen in de interviews met de drie agentexperten interessante bevindingen naar boven. Zo kan ik stellen dat de drie geïnterviewden het eens zijn over het feit dat JADE niet geschikt is voor agenttoepassingen voor de industrie. De redenen hiervoor kunnen we situeren op twee domeinen. Ten eerste kunnen we stellen dat de noden die in industriële toepassingen vervuld moeten worden vaak té specifiek zijn om te kunnen invullen met een platform als JADE. Wel moet duidelijk gesteld worden dat JADE ongetwijfeld goed zal werken voor een bepaalde set van toepassingen. Ook hier stemmen de resultaten uit de afzonderlijke interviews overeen. Het is namelijk zo dat ze JADE als een uitstekend platform definiëren voor onderzoek naar toepassingen van agenttechnologie en het simuleren van verantwoordelijkheden en architecturen van agenten. In brede zin is JADE uitstekend geschikt voor het afleveren van zogenaamde ‘proof-of-concepts’ waarmee niet gezegd wordt dat JADE niet nuttig kan zijn voor bepaalde specifieke toepassingen in de industrie.

Verder kunnen we formuleren dat de schaalbaarheidseisen aan een agentsysteem dat ontwikkeld wordt op basis van JADE niet overdreven mogen worden. Het blijkt immers uit ervaring dat het merendeel van de agenttoepassingen van JADE nog niet getest of

operationeel geweest zijn in een omgeving van meerdere platformen. Wanneer we ervan uitgaan dat JADE een tool is waarmee we 'proof-of-concepts' kunnen bewerkstelligen, is voorgaand negatief punt eigenlijk geen afbreuk aan JADE als agentplatform daar 'proof-of-concepts' op een niet al te grote schaal toegepast worden. Daardoor volstaat het dat men meerdere containers installeert op verschillende machines die dan behoren tot hetzelfde platform.

HOOFDSTUK 7: Praktijkcase SKF

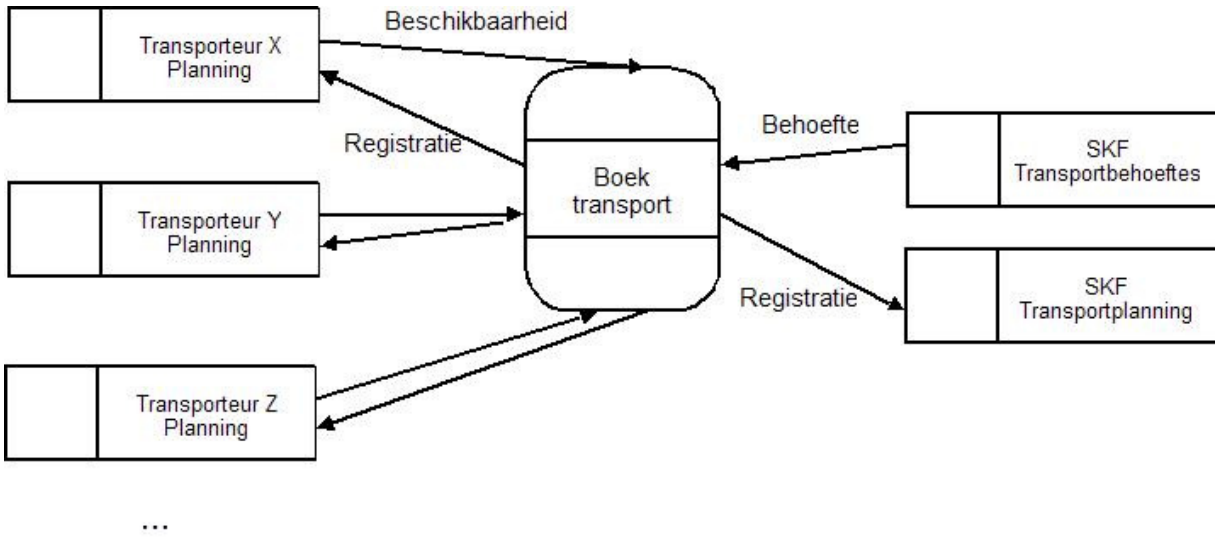
In deze case zullen we het gebruik van agenttechnologie nader toelichten via het doorlopen van enkele fasen in een levenscyclus van het ontwikkelen van een agentsysteem. Algemeen wordt aangenomen dat het ontwikkelen van een softwaresysteem bestaat uit vier fasen namelijk: selectie, analyse, ontwerp en productie. Aangezien dit eindwerk een exploratief onderzoek is van de toepassing van agenttechnologie binnen logistieke netwerken, zullen enkel de eerste twee fasen behandeld worden. Deze fases worden doorlopen om de abstracte concepten die tot nu toe aangehaald zijn te concretiseren om zo een inzicht te krijgen in bruikbaarheid, onvolkomenheden en mogelijkheden van agenttechnologie. De verwerkte informatie van het interview met de heer M. Linten is te vinden in bijlage IV.

7.1 Selectie

De situatie van SKF leent zich op basis van de inzichten, gepresenteerd in voorgaande hoofdstukken in dit eindwerk, uitstekend om het concept van agenttechnologie ten volle te benutten. Alhoewel nog andere processen typische kenmerken bezitten om ondersteund te worden door middel van een toepassing op agenttechnologie heb ik gekozen om het proces van het boeken van het 'domestic' transport nader te onderzoeken. Het 'domestic' transport heeft immers een flexibel karakter op gebied van frequentie, locatie van bestemming, vrachtvolume, vrachtmassa en laadeenheid. Dit komt doordat dit transport afgestemd moet worden op de specifieke eisen van de klant. In tegenstelling tot 'line-hauls' waar men probeert het volume en de massa te optimaliseren in functie van het laadvermogen van de gebruikte vrachtwagens, de laadeenheid altijd vaststaat en de locatie ook altijd gekend is. De specifieke eisen voor de laadeenheid zorgen er op dit moment zelfs voor dat SKF met verschillende transporteurs dient te werken voor het afhandelen van bepaalde 'domestic' transports.

In dit proces is er een intensieve communicatie met de transporteur nodig daar de stabiliteit van de 'line-hauls' hier niet terug te vinden is. Verder kunnen we stellen dat het proces om een vracht te boeken gedistribueerd van aard is omdat specifieke bedrijfsinformatie over de vracht die SKF dient te vervoeren gecommuniceerd moet worden naar de geschikte transporteur. Het is daarom dat agenttechnologie erg geschikt kan zijn om dit proces te ondersteunen.

De functionaliteit van het systeem dat uitgewerkt zal worden, bestaat er dus in om een vrachtbehoefte in te vullen door het boeken van een geschikt transport. Een data-flow diagram van bovenstaand proces vindt u in onderstaande figuur. In de figuur is duidelijk te zien dat informatie van verschillende ondernemingen samenkomt in het centrale proces, namelijk het boeken van een vracht.



Figuur 26: DFD van het agentsysteem

7.2 Analyse

Centraal in de analysefase staat het definiëren van de 'requirements' van het systeem. 'Requirements' zijn de mogelijkheden en voorwaarden waaraan het systeem moet voldoen. Van betreffende systemen kunnen we twee soorten vereisten definiëren namelijk de functionele vereisten en de niet-functionele vereisten. Dit wordt formeel gedefinieerd in het FURPS+ model. Naast de **F**unctionele vereisten zijn de niet-

functionele vereisten opgedeeld in '**U**sability', '**R**eliability', '**P**erformance' en '**S**ustainability'. (Grady, 1992: geciteerd in Eeles, 2005)

7.2.1 Functionele vereisten

Een populaire techniek om functionele vereisten van een systeem te modelleren is via zogenaamde 'use cases'. Via deze techniek worden 'use cases' opgebouwd die elk beschrijven hoe een bepaald doel behaald of een taak vervolledigd wordt. Deze verhalen over de verantwoordelijkheden van het systeem dienen eenvoudig en begrijpbaar te zijn voor alle belanghebbenden. (Cockburn, 2002) De vereisten van de mogelijke toepassing van agenten bij het domestic transport van SKF zullen we dus modelleren via het gebruik van deze use cases. Hiervoor werd een aangepaste versie van de beschikbare templates op usecases.org gebruikt, zoals weergegeven in bijlage V. (Use cases, 2007)

USE CASE 1	Verwerk vrachtorders	
Goal in Context	Database met open vrachtorders wordt gecontroleerd en als er nieuwe open orders komen worden deze geanalyseerd en verwerkt	
Scope	SKF Intern	
Preconditions	Systeem heeft operationele database-connectie	
Success End Condition	Het systeem heeft de nodige informatie verzameld over een open order om een transporteur te recruteren	
Failed End Condition	Het systeem heeft onvoldoende informatie over een bepaald open order.	
Primary, **Secondary Actors**	SKF agent Database met open orders	
Trigger	Geen	
DESCRIPTION	**Step**	**Action**
	1	SKF agent controleert database met open orders
	2	<herhaal voorgaande in een eeuwige lus>
	3	Nieuw open order verschijnt in database
	4	Agent haalt gegevens op
	5	Agent analyseert gegevens
	6	Agent schikt gegevens in formaat
EXTENSIONS	**Step**	**Branching Action**

	5a	Ontbrekende of foute gegevens 5a.1 Agent meldt fout in log 5a.2 Stop actie

USE CASE 2	Adverteren open order	
Goal in Context	Transporteurs worden op de hoogte gebracht van het nieuwe open order.	
Scope	SKF, transporteurs	
Preconditions	Er zijn geschikte en bereikbare transporteurs waarvan contactgegevens opgezocht kunnen worden	
Success End Condition	Transporteurs zijn op de hoogte van het open order.	
Failed End Condition	Transporteurs zijn niet op de hoogte van de open order.	
Primary, **Secondary Actors**	SKF agent Agenten transporteurs	
Trigger	Een geldig open order is verwerkt door de SKF agent.	
DESCRIPTION	**Step**	**Action**
	1	SKF agent zoekt contactgegevens van transporteuragenten op
	2	SKF agent maakt bericht verzendklaar
	3	SKF agent verstuurt bericht naar transporteuragenten
EXTENSIONS	**Step**	**Branching Action**
	1a	SKF agent is niet vertrouwd met formaat contactgegevens van een of meerdere transporteurs 1a.1 Schrijf melding weg naar log

USE CASE 3	Bezorg offerte
Goal in Context	Transporteuragenten analyseren en evalueren het ontvangen open order, maken antwoordbericht en versturen dit terug naar SKF agent
Scope	Intern transporteurs
Preconditions	De agenten van de transporteurs hebben bericht goed ontvangen, de agent heeft kennis over mogelijkheden en beperkingen van bedrijfsactiviteiten, agent heeft operationele connectie met planning.
Success End Condition	De voorwaarden en condities zijn gespecifieerd in een concrete offerte als antwoord op het open offer
Failed End Condition	De voorwaarden en condities van het open order kunnen niet ingevuld worden door de transporteur
Primary,	Agent transporteur

Secondary Actors	Database met planning van transporteur	
Trigger	Ontvangen van open order van SKF	
DESCRIPTION	**Step**	**Action**
	1	Transporteuragent verwerkt gegevens van ontvangen bericht
	2	Transporteuragent analyseert gegevens en beoordeelt haalbaarheid
	3	Transporteuragent raadpleegt planning
	4	Transporteur maakt offerte op *offerte kan ook weigering zijn
	5	Transporteuragent vervormt offerte in geschikt formaat
	6	Transporteuragent beantwoordt het open order via reply
EXTENSIONS	**Step**	**Branching Action**
	1a	Agent begrijpt het formaat van het bericht niet 1a.1 Schrijf melding weg naar log
	6a	Transporteuragent heeft geen contactgegevens over SKF agent 6a.1 Transporteuragent zoekt contactgegevens SKF agent 6a.2 Transporteuragent verzend bericht naar SKF agent

USE CASE 4	Boek transport	
Goal in Context	Een transport wordt geboekt en het open order wordt ingevuld. De transporteurs passen hun planning aan.	
Scope	SKF en transporteurs	
Preconditions	Er is minstens één geldige offerte teruggestuurd	
Success End Condition	Er is een transport geboekt	
Failed End Condition	Het open transportorder wordt niet ingevuld	
Primary, **Secondary Actors**	SKF transporteuragenten, database met open orders, database met planning	
Trigger	Afsluiten van indieningstijd	
DESCRIPTION	**Step**	**Action**
	1	SKF agent verzamelt inkomende offertes
	2	SKF agent rondt mogelijkheid tot kandidaatstelling voor open offerte af
	3	SKF agent evalueert offertes

	4	SKF agent aanvaardt de meest optimale offerte en boekt deze
	5	SKF agent deelt status offertes van transporteurs mee (aanvaarding of verwerping)
	6	Transporteuragenten passen planning aan
EXTENSIONS	**Step**	**Branching Action**
	3a	SKF agent detecteert ongeldige offerte 3a.1 SKF agent negeert offerte

7.2.2 Niet-functionele vereisten

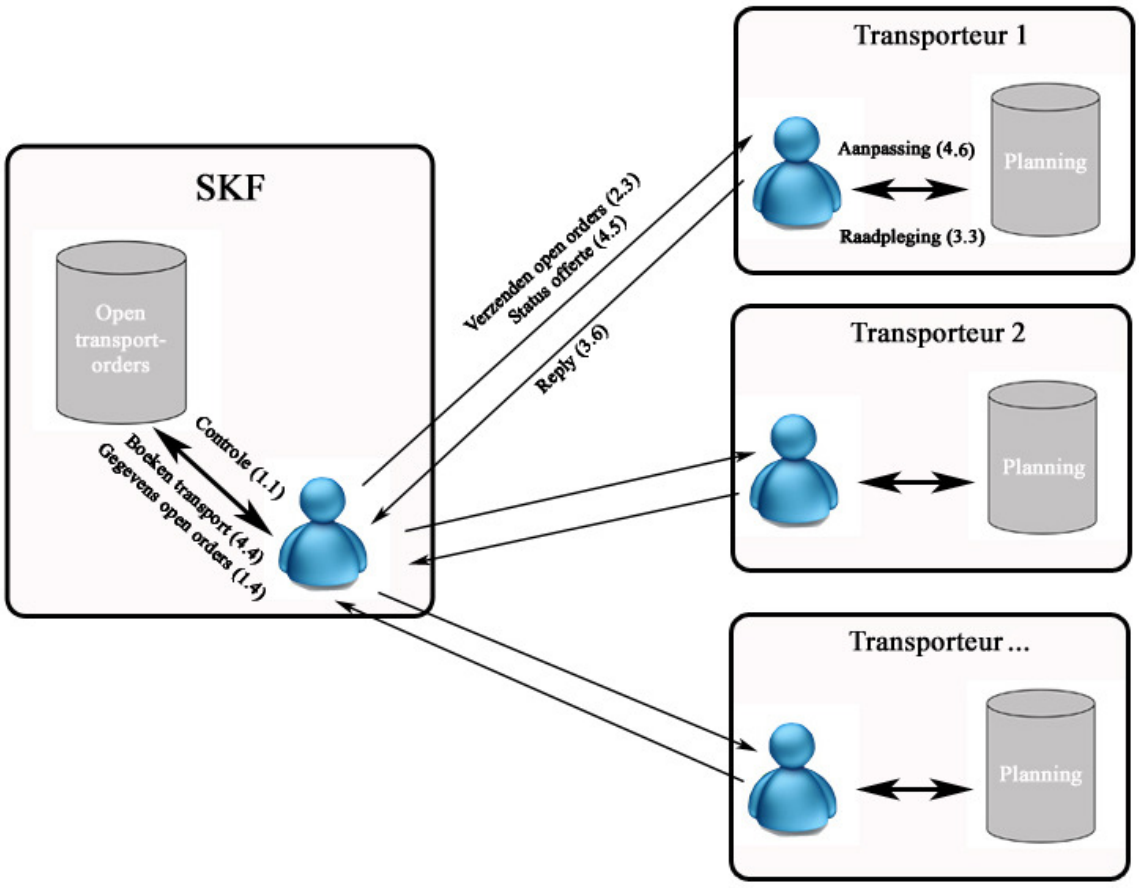
Niet-functionele vereisten kunnen moeilijk gemodelleerd worden via use cases daar zij opgebouwd zijn per doel dat we nastreven in een applicatie. Over de niet-functionele vereisten kunnen we stellen dat deze vereisten inherent zijn aan het gebruik van een multi-agentsysteem. Een multi-agentsysteem is immers opgebouwd uit verschillende componenten die los staan van elkaar en dynamische relaties met elkaar aanknopen. De 'reliability' wordt in die zin alleen aangetast als kritische componenten op systeemniveau aangetast worden. Verder kunnen we stellen dat via asynchrone procesafhandeling de middelen zoals hardware goed benut worden en de rekenkracht van het gehele systeem gegarandeerd is. Tot slot is de 'supportability' ook van een hoog niveau daar een agentsysteem dynamisch opgebouwd kan worden en elke entiteit als autonoom bestempeld wordt. Voorgaande beweringen zijn gefundeerd op de beschrijving van agentsystemen in hoofdstuk 3 en werden bekrachtigd in de afgelegde interviews.

7.2.3 Voorstelling

Zoals we reeds gezien hebben, zijn agentsystemen gedistribueerd en worden processen asynchroon behandeld via complexe interacties. Agenten kunnen reactief maar ook proactief verdrag vertonen. Verder is de opbouw van een agentsysteem dynamisch zodat de structuur en samenstelling niet eenduidig te definiëren is. Deze eigenschappen zijn de hoofdredenen die het modelleren van een agentsysteem ingewikkelder maken dan meer traditionele systemen. Hoewel de Unified Modeling Language (UML) stilaan geaccepteerd wordt als standaard voorstelling van softwaresystemen gebaseerd op het objectgeorienteerde paradigma, vertoont deze techniek tekortkomingen om

agentsystemen te modelleren. Het is ook daarom dat verschillende initiatieven op poten gezet zijn om specifieke modelleringtechnieken te ontwerpen voor agentsystemen. (Lynch en Rajendran, 2005; Bauer, 2002)

In de literatuur zijn vele voorstellen terug te vinden om de beperkingen van UML voor het modelleren van agenttoepassingen op te vangen gaande van interactiediagrammen, protocol- en architectuurdiagrammen. (Lynch en Rajendran, 2005) Een ander belangrijk initiatief probeert een uitbreiding te zijn op de bestaande UML namelijk Agent UML (AUML). (Bauer, 2002) Vermits deze case als illustratie dient van toepassingen op basis van agenttechnologie binnen logistieke netwerken beperken we ons echter tot het opmaken en bespreken van een architectuurdiagram.



Figuur 27: Architectuurdiagram agenttoepassing

7.3 Bespreking

De architectuur die we uitgetekend hebben is bijna een weerspiegeling van het agentplatform MamMoeT zoals dit ontwikkeld is door de vakgroep Informatietechnologie van de Katholieke Hogeschool Sint-Lieven te Gent. Deze mooie agenttoepassing werd ontwikkeld op basis van het JADE-platform en voorziet een systeem van transporteur- en verladeragenten die een één-op-één relatie vertonen met de logistieke spelers binnen een logistiek netwerk. Dit systeem is in staat om vrachten voor de binnenvaart op een autonome manier te negotiëren zonder menselijke betrokkenheid. In dit systeem werd bovendien een mechanisme geïmplementeerd dat rekening houdt met privacywensen van verladers en transporteurs. Dit mechanisme zorgt ervoor dat agenten de voorkeur geven om te onderhandelen met agenten die zij vertrouwen, in casu agenten waar zij veel transacties mee doen. Dit systeem dat gebouwd is als een 'proof-of-concept' is uitgebreid getest in verschillende situaties.

De voordelen die samengaan met deze toepassing zijn opgesomd in Bernaer et al. (2006). We kunnen deze voordelen toepassen op deze specifieke case, namelijk:

- Meer en snellere informatie over alternatieve vervoersmethoden zodat minder lege vrachten voorkomen en een snellere levertijd mogelijk is
- Goedkopere boeking van het transport door het vermijden van communicatiekosten
- Intelligente en flexibele informatiestroom die ten allen tijde up-to-date is
- Een flexibel, schaalbaar en gedistribueerd systeem dat een weerspiegeling is van het proces
- Makkelijk nieuwe transporteurs aansluiten op het systeem

Wel moeten we met de nodige voorzichtigheid omgaan met de voorgestelde architectuur. Het beslissingsproces van agenten dat we kunnen implementeren kan niet omgaan met erg complexe problemen. Zo is het moeilijk om agenten afwegingen laten te maken tussen verschillende parameters zonder dit expliciet te programmeren in een gedrag. Toch zijn in dit concrete voorbeeld de parameters die we dienen te negotiëren beperkt tot prijs en levertermijn. Andere parameters zoals de eenheid van lading, het volume en de massa van de lading en de locatie van de verlader en de klant zijn vaste parameters van de vracht.

HOOFDSTUK 8: Algemeen besluit en toekomstvisie

Aangezien in de loop van deze eindverhandeling reeds conclusies getrokken zijn na de afronding van elk logisch samenhangend gedeelte wordt in dit algemeen besluit een overzicht gegeven van deze conclusies en worden deze niet meer in diepte behandeld.

8.1 Overzicht van de belangrijkste conclusies

Op het einde van hoofdstuk 3 (3.6) vergeleken we de studie van de kenmerken en eigenschappen van agentsystemen met die van de studie over huidige toepassingen binnen de logistieke sector. Uit deze vergelijking leerden we dat we via het agentparadigma totaal verschillende toepassingen konden ontwikkelen waarvan de kenmerken uiterst geschikt zijn als ondersteuning van een dynamische, complexe en gedistribueerde omgeving.

In deze gedistribueerde en open omgeving is een belangrijke rol weggelegd voor standaarden die de mate van interoperabiliteit en integratie ten goede komen. De belangrijkste standaard, namelijk FIPA, definieert specificaties voor zowel de componenten waaruit een agentsysteem dient te bestaan als de interacties die agenten onderling en met deze componenten dienen te hebben.

Een conclusie die we op het einde van hoofdstuk 4 (4.8) trokken, is dat er nog bepaalde domeinen niet behandeld zijn door FIPA in de vorm van een standaard. Daarentegen zien we dat de FIPA-ACL en de gedefinieerde interactieprotocollen erg geschikt zijn om agentcommunicatie mogelijk te maken. Deze concepten kennen binnen de agentwereld dan ook een hoge graad van acceptatie. Verder lijkt het erop dat de FIPA standaarden die zich situeren op het gebied van de structuur die een agentsysteem dient te bekleden, voorbijgestoken worden door andere standaarden. Deze standaarden, waarvan de

belangrijkste Web Services, zijn ook ontwikkeld om de integratie en interoperabiliteit tussen software-entiteiten mogelijk te maken en vinden veel bijval in de industrie.

Het agentparadigma legt door zijn kenmerken bepaalde eisen op aan de bouw van zowel individuele agenten als het globale agentsysteem. Verder dienen agenttoepassingen gebruik te maken van onderliggende software-infrastructuren om bepaalde functionaliteiten mogelijk te maken binnen het agentsysteem. Deze eisen zorgen voor een grote complexiteit bij het ontwikkelen van agentsystemen en zijn een hoge drempel voor de ontwikkeling van agentsystemen. Een oplossing om deze drempels te vermijden is het gebruik van een agentplatform dat softwarebibliotheken bevat waar veel van deze functionaliteit reeds geïmplementeerd is. In deze context werd het op de FIPA-gebaseerde platform JADE besproken. Hoewel JADE erg geschikt is om snel en relatief eenvoudig te komen tot een werkende agenttoepassing conform de FIPA standaarden, formuleerden we op het einde van hoofdstuk 6 (6.9) onze bedenkingen bij het gebruik van dit platform als basis voor bedrijfstoepassingen. Zo is JADE uitstekend geschikt om proof-of-concept implementaties te realiseren waar de eisen op gebied van schaalbaarheid beperkt blijven.

Algemeen kunnen we stellen dat het gebruik van agentsystemen als ondersteuning voor logistieke netwerken een erg waardevolle gedachte is. Dit illustreerden we aan de hand van een concrete case in hoofdstuk 7. Toch hebben we kunnen vaststellen dat de toepassingsgraad van agenttechnologie binnen de bedrijfswereld nog op een laag pitje draait. Dit zal deels te verklaren zijn doordat het pragmatisch onderzoek naar agenttoepassingen nog niet volledig zijn maturiteit bereikt heeft en er nog veel ruimte is voor toekomstige ontwikkelingen. Een overzicht van deze verwachte evoluties wordt in volgende paragraaf gegeven.

8.2 Toekomstvisie

James Hendler (Hendler et al., 2007) roept de agentgemeenschap op tot reacties op zijn post in zijn weblog met als titel: “Where are all the agents?” Hij heeft het over de hype die agenttechnologie was einde jaren ’90 en over het geloof in ware integratie en coöperatie die men in de toekomst kon bewerkstelligen door middel van agenten. Toch zijn de toepassingen volgens hem tot op de dag van vandaag eerder in academische

milieus te situeren en niet in de bedrijfswereld. Ook prof. dr. Holvoet is het ermee eens dat de gerealiseerde agenttoepassingen niet in verhouding zijn tot de inspanningen op gebied van het uitgevoerde onderzoek.

Toch kunnen we uit de interviews en de reacties op de weblog (Hendler et al., 2007) leren dat er geen reden is om aan te nemen dat toepassingen van agenttechnologie nooit gerealiseerd gaan worden. We zien immers dat steeds meer kenmerken van agenttechnologie doordringen in nieuwe softwaretoepassingen. Het komt er dus op neer dat onderzoek naar het paradigma van agenttechnologie verschillende deuren opengezet heeft voor toepassingen die meer waarde kunnen creëren voor de bedrijfswereld op een gebruiksvriendelijke manier.

Zo vermeldt het rapport van Agentlink III (2005) dat onder andere de trends van autonomic computing en complex systems zich op dit moment afspelen in de softwarewereld. 'Autonomic computing' slaat logischerwijs op de steeds stijgende mate van autonomie die software krijgt. Verder zijn moderne software en technologische systemen steeds complexer aan het worden. Op deze manier wordt de complexiteit van de realiteit weerspiegeld in software die op deze manier ook een betere ondersteuning kan bieden. Zowel het omgaan met de complexiteit als de mate van autonomie binnen agentsystemen werd uitvoerig behandeld in deze eindverhandeling. We kunnen dus besluiten dat kenmerken en concepten van agenttechnologie zich meer en meer zullen manifesteren binnen softwaretoepassingen. Toch valt het in twijfel te trekken of deze toepassingen door het leven zullen gaan onder de noemer 'agenttoepassing'.

In deze eindverhandeling werd al vermeld dat er naast de FIPA-standaarden veelbelovende standaarden hun opgang vinden die een interoperabiliteit en herbruikbaarheid van softwarecomponenten bewerkstelligen. Hoewel FIPA dus streeft om de standaard van agenttoepassingen te worden via de specificatie van haar referentiemodel voor agentplatformen, is het dus verre van zeker of deze trend zich in de toekomst zal verderzetten. We moeten wel vermelden dat FIPA naast het specificeren van standaarden ook bijdraagt aan het vormgeven van het gedachtegoed dat hoort bij agenttechnologie. Zo kunnen we stellen dat FIPA als een belangrijk discussieforum kan gezien worden om te komen tot meer functionele softwarearchitecturen en toepassingen die het gebruikersgemak verhogen.

Ook de experten die ik interviewde zijn erg sceptisch over de FIPA-standaarden betreffende structuur en interactie op een agentsysteem. Zij verwachten ook dat in de toekomst applicaties voor het bedrijfsleven eerder via andere standaarden zullen werken. Vooral inspanningen op het gebied van Web Services en Semantic Web zijn snel in opgang gekomen. In volgende paragrafen maken we duidelijk dat de opkomst van deze standaarden niet als een bedreiging maar eerder als een verrijking kan dienen voor toepassingen op het gebied van agenttechnologie.

Zo kunnen Web Services gezien worden als een infrastructuur die bijna ideaal is om de opbouw van agentsystemen en de interacties tussen deze softwareagenten te realiseren. Verder kunnen we stellen dat de toepassingen van Web Services enorm groeien en ondersteund worden door toepassingen in het bedrijfsleven. (Agentlink III, 2005) Dit komt door de doordachte infrastructuur en het gemak waarmee toepassingen interoperabel kunnen zijn. Aan de andere kant kunnen we stellen dat Web Services verrijkt kunnen worden door agenten. Agenten zijn immers applicaties met empowerment die iets of iemand vertegenwoordigen en die gericht acties ondernemen om hun doel te bereiken. Deze acties zouden dan inhouden dat agenten Web Services aanvragen die geïmplementeerd worden door een andere agent. Deze zienswijze wordt door de een werkgroep van FIPA die samenwerkt met het IEEE bekrachtigd (W3C Web Services Architecture Working Group Note, 2004: geciteerd in IEEE-FIPA SC, 2007): "In effect, software agents are the running programs that drive Web services – both to implement them and to access them."

De uitbouw van het Semantische Web zoals we dat kort besproken hebben in hoofdstuk 5 vereist het aanwezig zijn van software-entiteiten die in staat zijn om de semantische informatie te interpreteren. We kunnen dus stellen dat het Semantische Web een fundament kan zijn voor agenttoepassingen die informatie kunnen interpreteren, lokaliseren, beheren en uitwisselen. (Agentlink III, 2005) In het rapport van Agentlink III (2005) vinden we zelfs terug dat de agentwereld betrokken dient te zijn met de ontwikkeling van het Semantic Web daar deze twee onderzoeksdomeinen nauw verbonden zijn met elkaar.

Literatuurlijst

- AgentLink III (2004) "Agent Technology Roadmap: Overview and Consultation Report", (online) (geraadpleegd op 1 oktober 2006) <URL: http://www.agentlink.org/roadmap/roadmapreport.pdf>

- Agentlink III (Luck M., McBurney P., Shehory O., Willmott S. and the AgentLink Community) (2005) "Agent Technology Roadmap: A Roadmap for Agent Based Computing", (online) (geraadpleegd op 2 oktober 2006) <URL: http://www.agentlink.org/roadmap/al3rm.pdf>

- Bauer, B. (2002) "UML class diagrams revisited in the context of agent-based systems", in: Wooldridge, M., Weiß, G., Ciancarini, P. (Eds.) (2002), "Agent-Oriented Software Engineering II", Volume 2222, p. 101-118, Heidelberg: Springer

- Bellifemine, F., Caire, G., Poggi, A. en Rimassa, G. (2003) "JADE, a whitepaper". (online) (geconsulteerd op 30 november 2006) <URL: http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf>

- Bernaer, S., Burke, E., De Causmaecker, P., Vanden Berghe, G. en Vermeulen, T. (2006), "A multi agent system to control complexity in multi modal transport", (online) (geraadpleegd op 22 september 2006) <URL: http://www.asap.cs.nott.ac.uk/publications/pdf/MammoetCisRam06.pdf>

- Bitran, G., Gurumurthi, S., en Lin Sam, S. (2006) "Emerging Trends in Supply Chain Governance", MIT Sloan School of Management. (online) (geraadpleegd op 15 december 2006) <URL: http://ebusiness.mit.edu/research/papers/227_Sam_Emerging_Tends_Supply_Chain_Goverance.pdf>

- Brunn, P., Jensen, M. en Skovgaard, J. (2002) "e-Marketplaces: Crafting a winning strategy", European Management Journal, Vol. 20, n° 3, p. 266-298

- Chaib-Draa, B. en Dignum, F. (2002) "Trends in agent communication language", Computational Intelligence, Vol. 18, n° 2

- Chase, R., Jacobs, R. en Aquilano, N. (2006) "Operations management for competitive advantage", New York: McGraw-Hill

- Chen, L. en Sycara, K. (1997) "WebMate : A Personal Agent for Browsing and Searching", (online) (geraadpleegd op 26 oktober 2006) <URL: http://www.ri.cmu.edu/pub_files/pub1/chen_liren_1998_1/chen_liren_1998_1.pdf>

- Christopher, M. (2000) "The Agile Supply Chain Competing in Volatile Markets", Industrial Marketing Management, vol. 29, n°1, p.37–44

- Cockburn, A. (2002) "Use cases, ten years later", (online) (geraadpleegd op 5 mei 2007) <URL: http://alistair.cockburn.us/index.php/Use_cases%2C_ten_years_later>

- Coebergh, P. (1999) "Intermodaal vervoer, een praktisch handboek voor de manager", Utrecht: Lemma BV

- Decker, K. en Mersic, M. (2001) "DECAF - A Flexible Multi Agent System Architecture", Amsterdam: Kluwer Academic Publishers

- Delsupehe, C. (2005) "Standaarden voor Web Services, een overzicht". Eindverhandeling Informatica, Universiteit Hasselt, Diepenbeek

- Depaire, B. (2003) "Een vergelijkende studie van ontwikkelingsplatformen voor Web Services", Eindverhandeling Handelsingenieur Beleidsinformatica, Limburgs Universitair Centrum, Diepenbeek

- Dickinson, I. (2004) "What future for FIPA?", Reactive, autonomous, 15-11-2004, (online) (geraadpleegd op 21 mei 2007) <URL: http://nuin.blogspot.com/2004_11_01_archive.html>

- Dragoni, N., Gaspari, M. en Guidi, D. (2006) "An infrastructure to support cooperation of knowledge-level agents on the semantic Grid", Springer science and Business Media, LLC.

- Eeles, P. (2005) "Capturing Architectural Requirements", (online) (geraadpleegd op 14 mei 2007) <URL: http://www-128.ibm.com/developerworks/rational/library/4706.html>

- Etzioni, O. en Weld, D. (1994) "A Softbot-Based Interface to the Internet". Communications of the ACM, 1994, Vol. 37, n° 7, p. 72-76

- Evangelista, P. (g.d.) "Understanding ICT management in small transport and logistics service providers", Indian Institute of Materials Management, (online) (geraadpleegd op 5 december 2006) <URL: http://www.iimm.org/knowledge_bank/IFPSM/Pietro%20Evangelista.pdf>

- FAMAS.MV2 (2003) "Eindrapport Truck Afhandeling", (online) (geraadpleegd op 12 oktober 2006) <URL: http://www.initi8.org/nl/files/documents/Eindrapport%20FAMAS%20MV2%20TA%20definitief.pdf>

- Finin, T., Labrou, Y. en Mayfield, J. (1994) "KQML as an agent communication language". (online) (geraadpleegd op 3 november 2006) </URL: http://www.cs.umbc.edu/kqml/papers/kqmlacl.pdf>

- Finin, T., Labrou, Y. en Mayfield, J. (1995) "KQML as an agent communication language", in: Bradshaw , J. (Eds) (1995), "Software agents", Cambridge: MIT Press

- FIPA Website (2007a) (online) (geconsulteerd op 2 april 2007) <URL: http://www.fipa.org>

- FIPA Website (2007b) "Paper submitted to AAMAS 2007". (online) (geraadpleegd op 25 mei 2007) <URL: http://www.fipa.org/subgroups/AWSI-WG.html>

- FIPA Website (2007c) "IEEE-FIPA (pre-)meeting at AAMAS 2005-07-27", (online) (geraadpleegd op 22 mei 2007) <URL: http://www.fipa.org/subgroups/ROFS-SG-docs/FIPA-Review-At-AAMAS-2005-07.htm>

- Franklin, S. en Graesser, A. (1996) "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents", Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages. (online) (geraadpleegd op 16 november 2006) <URL: http://www.msci.memphis.edu/~franklin/AgentProg.html>

- Gelati, J. (2004) "Agent-based systems in Public Administrations". International Review of Law Computers & Technology, vol. 18, n° 1, p. 97–107.

- Greenwood, D. (2005) " JADE Web Services Integration Gateway". (online) (geraadpleegd op 18 mei 2007) <URL: http://x-opennet.org/netdemo/Demos2005/aamas2005_netdemo_2pg.pdf>

- Gruber, T. (1993) "A Translation Approach to Portable Ontology Specifications", Academic Press, June 1993

- Gunasekaran, A. en Ngai, E. (2003) "Information systems in supply chain integration and management", European Journal of Operational Research, n° 159, p. 269 – 295.

- Hendler, J. en reacties (2007) "Where are all the agents", Mindswap weblog. (online) (geraadpleegd op 1 mei 2007) <URL: http://www.mindswap.org/blog/2007/04/23/where-are-all-the-agents-long-form/>

- Hermans, B. (1996) "Intelligent Software Agents on the Internet: an inventory of currently offered functionality in the information society & a prediction of (near-)future developments". (online) (geraadpleegd op 14 oktober 2006) <URL: http://www.hermans.org/agents>

- IEEE-FIPA SC "Agents and Web Services Interoperability Working Group, Charter Proposal." (online) (geraadpleegd op 21 mei 2007) <URL: http://www.fipa.org/subgroups/AWSI-WG-docs/AWSI_WG_Charter.pdf>

- JADE board (2006a) "JADE programmer's guide", (online) (geraadpleegd op 1 februari 2007) <URL: http://jade.tilab.com/doc/programmersguide.pdf >

- JADE board (2006b) "JADE administrator's guide", (online) (geraadpleegd op 1 februari 2007) <URL: http://jade.tilab.com/doc/administratorsguide.pdf>

- JADE Website (2007a) (online) (geconsulteerd op 12 maart 2007) <URL: http://jade.tilab.com/>

- JADE Website (2007b) "Section: Frequently Asked Questions (FAQ)", (online) (geconsulteerd op 12 maart 2007) <URL: http://jade.tilab.com/>

- Jennings, N. en Wooldridge, M. (1994) "Intelligent agents: Theory and Practice", Knowledge Engineering Review, October 1994

- Jennings, N., Sycara, K., en Wooldridge, M. (1998) "A roadmap of agent research and development", Autonomous Agents and Multi-Agents Systems, Vol. 1, p. 7-38.

- Keele, J. en Wray, J. (2005) "Software agents in molecular computational biology", Briefings in bioinformatics, Vol. 6, n° 4, p. 370-379

- Korba L., (2000) "Towards an agent middleware framework for e-commerce". Netnomics 2 (2000) 171–189 171

- Kotler, P. (2003) "Principes van Marketing", Amsterdam: Pearson Education

- Labrou, Y., Finin, T., Peng, Y. (1999) "Intelligent Agents. Agent Communication Languages: the current landscape", IEEE Intelligent Systems, (online) (geraadpleegd op 7 mei 2007) <URL: http://www.flacp.fujitsulabs.com/~yannis/publications/ieeeIntelligentSystems1999.pdf>

- Langley, N. (2003) "Great technologies for 2004", Computer Weekly, 12/2/2003, p. 36-37.

- Laudon, K. en Laudon, J. (2005) "Management Information Systems: managing the digital firm", New Jersey: Pearson Prentice Hall

- Lostwax, (2007) (online) (geraadpleegd op 27 april 2007) <URL: http://www.lostwax.com>

- Luck, M. McBurney, P. en Preist C. (2004) "A Manifesto for Agent Technology: TowardsNext Generation Computing", Autonomous Agents and Multi-Agent Sytems, vol. 9, p. 203–252.

- Lynch, S. en Rajendran, K., (2005) "Multi-agent systems design for novices", Computer Science Education, Vol. 15, n° 1, p. 41-57.

- Macharis, C. (2004) "Coöperatie of concurrentie? De intermodale transportketen nader bekeken", in: Blauwens, G., d'Haens, P., Van Breedam, A., Logistiek: laatste front in de concurrentieslag. Vlaams-Wetenschappelijk Economisch Congres, 25 – 26/03/04. Garant: Antwerpen

- Matthyssens, P., Martens, R., en Vandenbempt, K. (1998) "Concurrentiestrategie en marktdynamiek : op weg naar concurrentievoordeel in industriële markten", Deventer: Kluwer

- Morgan, B. (1997). "Corba meets Java". (online) (geraadpleegd op 13 april) <URL: http://www.javaworld.com/cgi-bin/mailto/x_java.cgi>

- Nassen, K. (2006) "Met ERP inspelen op de huidige uitdagingen in de ondernemingswereld", Eindverhandeling Handelsingenieur operationeel management en logistiek, Universiteit Hasselt, Diepenbeek

- Nederland Logistiek Netwerkland (2006), "Mogelijkheden voor gebruik van innovatieve ICT-toepassingen in Transport & Distributie in 2006", (online) (geraadpleegd op 13 oktober 2006) <URL: http://www.syntens.nl/NR/rdonlyres/45244F85-D40C-408A-8F64-139D9022B84C/398/NederlandLogistiekNetwerkland1.pdf>

- Nolan R. en Mcfarlan W. (2005) "Information Technology and the board of directors", Harvard Business Review, October 2005 p. 96-106.

- Noordzij, M. (g.d.) "Web Services", (online) (geraadpleegd op 21 mei 2007) <URL: http://www.2en40.nl/ws_itmonitor.html>

- Nwana, H. (1996) "Software agents: an overview", Knowledge Engineering Review, Vol.11, n° 3, p. 1-40

- Odell, J. (2002) "Objects and Agents Compared", Journal of Object Technology, vol. 1, n°1, p. 41-53

- Omair, S., Arshad, A., Hiroki, S. en Farooq, A. (2007) "Analysis of existing AWSI related implementations. AgentWeb Gateway for dynamic and seamless integration of FIPA Multi Agent Sysem and W3C Web Service System." (online) (geraadpleegd op 21 mei 2007) <URL: http://www.fipa.org/subgroups/AWSI-WG-docs/Omair_AWSI_Req_Doc.doc>

- Omicini, A. en Rimassa, G. (2004) "Towards seamless agent middleware", (online) (geraadpleegd op 30 november 2006) <URL: http://www.soclab.bth.se/workshops/tapocs2004/downloads/3-1_rimassa.pdf>

- Palmer, B. (2001) "The Semantic Web: An Introduction", (online) (geraadpleegd op 1 mei 2007) <URL: http://infomesh.net/2001/swintro/>

- Panko, R. (2005) "Datanetwerken en telecommunicatie", 5e editie, Amsterdam: Pearson Prentice Hall

- Papazoglou, M. (2001) "Agent-oriented technology in support of e-business". Communications of the ACM, 2001, Vol. 44, n° 4, p. 71-77

- Poggi A., Rimassa G. en Turci P. (2002) " What Agent middleware can (and should) do for you", Applied Artificial Intelligence, Vol. 16, p. 677-698

- Poggi, A. (1999) "Developing Real Applications with agent technologies", Journal of systems integration, Vol. 9, p. 311-328.

- Poslad, S., Buckle, P. en Hadingham, R. (2001). "Open source, standards and scaleable agencies, in infrastructure for agents", p. 296-303 in: Wagner, T. en Rana O. (2001) "Infrastructure for agents", Heidelberg: Springer-Verlag

- PwC (PricewaterhouseCoopers) consulting (2002) "Opportunities and Challenges for Logistics Service Providers in Europe" (online) (geraadpleegd op 17 november 2006) <URL: http://www.eyefortransport.com/ResArticles/logistics_eng.pdf>

- Ranganathan, C., Dhaliwal, J., en Teo,T. (2004) "Assimilation and Diffusion of Web Technologies in Supply-Chain Management: An Examination of Key Drivers and Performance Impacts", International Journal of Electronic Commerce, Vol. 9, n° 1, pp. 127–161

- Rao, A. en Georgeff, M. (1995) "BDI Agents: from theory to practice", in: ICMAS (1995), "Proceedings of the First International Conference on Multi-Agent Systems", p. 312–319, SanFrancisco: CA

- Reilly, D. (2006) "Java RMI & CORBA. A comparison of two competing technologies". (online) (geraadpleegd op 13 april 2007) <URL: http://www.javacoffeebreak.com/articles/rmi_corba/>

- Rome, F. (2004) "Globalisatie en logistiek: trends, uitdagingen en opportuniteiten", in: Blauwens, G., d'Haens, P., Van Breedam, A., Logistiek: laatste front in de

concurrentieslag. Vlaams-Wetenschappelijk Economisch Congres, 25 – 26/03/04. Garant: Antwerpen.

- Semantic Web Agreement Group (2001) "What is the Semantic Web?" (online) (geraadpleegd op 1 mei 2007) <URL: http://swag.webns.net/whatIsSW>

- Singh, M. (1998) "Agent communication languages: rethinking the principles". (online) (geconsulteerd op 12 april 2007) <URL: http://www.csc.ncsu.edu/faculty/mpsingh/papers/mas/computer-acl-98.pdf>

- SKF.com (online) (geraadpleegd op 11 mei 2007) <URL : www.skf.com>

- Sultan, A. en Nien, S. (2006) "Safe Credential-Based Trust Protocols: A Framework", IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings)(WI'06), p. 949-952

- Sun Website, "Remote method invocation home", (online) (geraadpleegd op 12 april 2007) <URL: http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp>

- Szirbik, N. en Wortmann, H. (2001) "ICT issues among collaborative enterprises: from rigid to adaptive agent-based technologies", Production planning and control, Vol. 12, n° 5, p. 452-465

- UMBC AgentWeb (2007a) "KIF, knowledge interchange format". (online) (geraadpleegd op 8 mei 2007) <URL: http://www.csee.umbc.edu/kse/kif/>

- Use cases (online) (geraadpleegd op 8 mei 2007) <URL: http://www.usecases.org>

- van Aart, C. (2001) "Intelligent Agents: korte introductie en demo", ISOC bijeenkomst Zoetermeer, (online) (geraadpleegd op 26 oktober 2006) <URL: http://isoc.nl/activ/documenten/2001-261101-ChrisVanAart.ppt>

- Van Toor, F. (2004) "Logistiek vastgoed", (online) (geraadpleegd op 13 oktober 2006) <URL: http://static.realworks.nl/cms/30068/Toor-Masterproof.pdf>

- Venners, B. (1996) "The lean, mean, virtual machine". (online) (geraadpleegd op 1 december 2006) </URL: http://www.javaworld.com/javaworld/jw-06-1996/jw-06-vm.html>

- Verduijn, T., Becker, J., van Hillegersberg, J., van de Velde, S., Moonen, H. en Huijsman, M. (2003) "Intelligent Agents in supply chains, an explorative research", (online) (geraadpleegd op 15 december 2006) <URL: http://www.hansmoonen.com/publications/2003-connekt_ia.pdf>

- Vulkan, N. (1999) "Economic Implications of agent technology on e-commerce", The Economic Journal, Vol. 109 (February), p. 67-90

- W3C (2004) "OWL Web Ontology Language, Overview", (online) (geraadpleegd op 1 mei 2007) <URL: http://www.w3.org/TR/2004/REC-owl-features-20040210/#s1.1>

- W3Schools (2007) "RDF", (online) (geraadpleegd op 1 mei 2007) <URL: http://www.w3schools.com/rdf>

- Whitestein Technologies AG , (2005) "JADE WEB SERVICES INTEGRATION GATEWAY (WSIG) GUIDE ". (online) (geraadpleegd op 14 april 2007) <URL: http://jade.tilab.com/doc/tutorials/JADE_WSIG_Guide.pdf>

- Wikipedia (2007) "Web service" (online) (geraadpleegd op 21 mei 2007) <URL: http://en.wikipedia.org/wiki/Web_service>

Bijlagen

Bijlage I: Bespreking van de rollen in een logistiek netwerk

(Macharis (2004), FAMAS.MV2 (2003) en Coebergh (1999))

Verlader

De verlader geeft opdracht tot het vervoeren van een lading aan een rederij, expediteur, operator of wegvervoerder. Voor de verlader is het van belang dat hij zo goed mogelijk op de vraag van zijn klant, de ontvanger, inspeelt en dat de goederen derhalve op tijd en in de juiste hoeveelheid en staat worden afgeleverd. Om dit te realiseren heeft de verlader een informatiebehoefte voor wat betreft de status van zijn zending.

Expediteur

De expediteur organiseert op verzoek van de verlader het vervoer en de afhandeling van de bijkomende formaliteiten (boekingen, douaneformaliteiten en de vereiste administratie). De expediteur plant derhalve het transport, schakelt de noodzakelijke partijen in en geeft feedback aan de verlader over hoe de stand van zaken is omtrent de verzending. Ook voor de expediteur is het van belang zo goed mogelijk op de vraag van zijn klant, de verlader, te anticiperen. Het is daarbij belangrijk over actuele statusinformatie te beschikken. De expediteur wordt afgerekend op het aantal zendingen waarvoor hij zorg draagt en hij heeft er belang bij dit tegen minimale kosten te doen. De expediteurs zijn in feite logistieke organisatoren.

Rederij

De rederij is de meestal de eigenaar van de schepen en containers en is verantwoordelijk voor het transport over zee. Ze baat een (regelmatige) lijn of dienst van of naar bepaalde bestemmingen uit en wordt daartoe vooral ingeschakeld door de expediteur. de term die men voor deze manier van werken hanteert is ‘merchant haule’. In tegenstelling tot de ‘merchant haule’ ontwikkelen rederijen ook meer en meer spoor- en binnenvaartdiensten van en naar het achterland. Deze activiteit wordt ‘carriers haule’

genoemd en betekent dus dat de rederij ook kan fungeren als aanspreekpunt voor de verlader om de logistieke keten deur-tot-deur te organiseren.

De rederij wordt voornamelijk beoordeeld op leverbetrouwbaarheid, wat neerkomt op een tijdige en onbeschadigde levering. Daarnaast wordt de rederij beoordeeld op flexibiliteit: hoe kort voor de afvaart kan nog geboekt worden en hoe laat moet de lading uiterlijk aangeleverd worden? De rederij heeft belang bij het maximaliseren van de beladingsgraad van het schip, het minimaliseren van de ligtijd van een schip en het maximaliseren van de betrouwbaarheid hiervan. Hierbij is het belangrijk dat de containers op het juiste tijdstip op de juiste plaats zijn en dat het herpositioneren van deze containers zo efficiënt mogelijk gebeurt.

Operator

De operatoren bieden binnen de keten bepaalde logistieke diensten aan. De klanten van de operators kunnen verschillende rolspelers in de keten zijn zoals de verlader, de expediteur, de (zee)rederij of een andere operator.

De binnenvaartoperator en railoperator zijn logistieke dienstverleners gericht op het goederenvervoer tussen twee terminals. Meestal is dit voor de binnenvaartoperator een zeehaven en een inlandterminal waartussen de operators binnenschippers de goederen via de waterwegen laten vervoeren. De barge-operator is een binnenvaart- of railoperator die naast het gebruikelijke takenpakket ook nog de overslag van de containers en het voor- en natransport van de goederen regelt. Voor het vervoer via de binnenvaart wordt er een binnenschipper ingeschakeld. Over de railoperatoren zegt Macharis (2004) het volgende: “ze organiseren vaste diensten met pendeltreinen vanaf de spoor/weg terminals, bepalen de prijs van het traject in samenspraak met de spoorwegmaatschppij en zijn eigenaar of huurder van de wagons”. Voor het voor- en natransport schakelt de operator, rederij, expediteur, terminaluitbater of de verlader zelf, afhankelijk van de gekozen optie, een wegtransporteur in.

De operators streven naar een optimale bezetting van hun transportmodi. De operators hebben belang bij een snelle behandeling van hun lading en een goede afstemming van het laden en lossen bij de terminals.

Wegvervoerder/transporteur

De wegvervoerder, ingeschakeld door de expediteur, transporteert de lading per vrachtwagen tussen verladers/ontvangers, terminals en lege depots. Hij heeft als primair belang zo efficiënt mogelijk te transporteren en de afspraken met de expediteur zo goed mogelijk na te komen. In meer operationele zin wil de wegvervoerder de verblijftijd van vrachtwagens op het terminalterrein minimaliseren en met een hoge beladingsgraad opereren. De wegvervoerder kan ook opereren op zelfstandige basis zodat zijn dienst niet kadert in een voor- of natransport. Hij wordt dan een rechtstreeks aanspreekpunt van de verlader.

Terminaloperator

De terminaloperator laadt en lost transportmodi op een overslagplaats of terminal. Er bestaan drie terminals namelijk de maritieme, de spoor- en de binnenvaartterminal. Een trend die meer en meer optreedt is dat belangrijke binnenvaart- en railoperatoren hun eigen terminals bouwen en dus het laden en lossen als standaardpakket in hun logistieke dienst opnemen. De criteria waarop een terminal wordt beoordeeld zijn de prijs per geladen/geloste container of vracht, de los-/laadsnelheid (het aantal geladen/geloste containers per uur of tijdsduur lossen/laden specifieke vracht), de kosten van het verblijf van goederen op de terminal (opslag) en de mogelijkheden tot opslag van lege containers op de terminal. De uitbater van de terminal zal altijd proberen om zijn overslagcapaciteit optimaal te benutten. Om dit te kunnen doen is het onder meer noodzakelijk om over actuele en betrouwbare informatie te beschikken over de lever- en ophaaltijden van de goederen

Bijlage II: Een overzicht van de FIPA standaarden

(http://www.fipa.org/specs/index.html)

Identifier	00023
Title	FIPA Agent Management Specification
Source	FIPA TC B
Type	Component
Subject	Management
Summary	Management for agents on FIPA agent platforms.
Keywords	FIPA,2000,Standards,Agents,Agent Management,Agent Platforms,Agent Environments,Service Descriptions
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2004-03-18	Standard	SC00023K.html, SC00023K.pdf	K
	2002-12-06	Standard	SC00023J.html, SC00023J.pdf	J
	2002-10-31	Experimental	XC00023I.pdf, XC00023I.html	I
	2001-08-15	Experimental	XC00023H.html, XC00023H.pdf	H
	2000-08-28	Experimental	XC00023G.pdf, XC00023G.html	G
	2000-11-30	Experimental	XC00023F.pdf, XC00023F.html	F

Identifier	00014
Title	FIPA Nomadic Application Support Specification
Source	FIPA TC E
Type	Informative
Subject	Applications
Summary	A description of agents and an ontology for supporting nomadic applications and devices.
Keywords	FIPA,2000,Standards,Agents,Applications,Wireless Networks,Wireless Devices,Nomadic Applications
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SI00014H.html, SI00014H.pdf	H
	2002-10-31	Experimental	XI00014G.pdf, XI00014G.html	G
	2002-10-31	Experimental	XI00014F.pdf, XI00014F.html	F
	2002-10-31	Experimental	XI00014E.pdf, XI00014E.html	E
	2001-08-15	Experimental	XC00014D.html, XC00014D.pdf	D

2000-08-17	Experimental	XC00014C.pdf, XC00014C.html	C
2000-11-30	Experimental	XC00014B.pdf, XC00014B.html	B
2000-11-30	Experimental	XC00014A.pdf, XC00014A.html	A

Identifier	00091
Title	FIPA Device Ontology Specification
Source	Gateways TC
Type	Informative
Subject	Ontologies
Summary	An ontology for describing devices and their properties
Keywords	FIPA,2000,Standards,Agents,Ontology,Devices,Profiles
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SI00091E.html, SI00091E.pdf	E
	2002-10-31	Experimental	XC00091D.pdf, XC00091D.html	D
	2002-04-22	Experimental	XC00091C.html, XC00091C.pdf	C
	2001-08-15	Preliminary	PC00091B.html, PC00091B.pdf	B
	2001-04-12	Preliminary	PC00091A.pdf, PC00091A.html	A

Identifier	00094
Title	FIPA Quality of Service Specification
Source	FIPA TC Nomadic Application Support
Type	Component
Subject	Applications
Summary	A definition of an ontology describing message transport Quality of Service for FIPA.
Keywords	FIPA,2000,Standards,Agents,Message Transport,Quality of Service,Ontologies
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00094A.html, SC00094A.pdf	A
	2002-10-31	Experimental	XC00094.pdf, XC00094.html	Initial

Identifier	00088
Title	FIPA Agent Message Transport Envelope Representation in Bit Efficient Specification
Source	Gateways TC
Type	Component

Subject	Envelope Representations
Summary	A description of a message transport envelope representation in a bit efficient encoding.
Keywords	FIPA,2000,Standards,Agents,Message Envelope,Envelope Representation,Bit Efficient Encoding
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00088D.html, SC00088D.pdf	D
	2002-10-31	Experimental	XC00088C.pdf, XC00088C.html	C
	2001-08-15	Experimental	XC00088B.html, XC00088B.pdf	B
	2001-08-08	Experimental	XC00088A.pdf, XC00088A.html	A
	2000-10-16	Preliminary	PC00088.pdf, PC00088.html	Initial

Identifier	00085
Title	FIPA Agent Message Transport Envelope Representation in XML Specification
Source	Transport WG
Type	Component
Subject	Envelope Representations
Summary	A description of a message transport envelope representation in an XML encoding.
Keywords	FIPA,2000,Standards,Agents,Message Envelope,Envelope Representation,XML Encoding
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00085J.html, SC00085J.pdf	J
	2002-10-31	Experimental	XC00085I.pdf, XC00085I.html	I
	2001-08-15	Experimental	XC00085H.html, XC00085H.pdf	H
	2000-08-17	Experimental	XC00085G.pdf, XC00085G.html	G
	2000-11-30	Preliminary	PC00085F.pdf, PC00085F.html	F

Identifier	00084
Title	FIPA Agent Message Transport Protocol for HTTP Specification
Source	Transport WG
Type	Component
Subject	Transport Protocols
Summary	A description of a message transport protocol based on HTTP.

Keywords	FIPA,2000,Standards,Agents,Message Transport Service,MTS,Network Transport,Message Delivery,Network Protocols,HTTP
Superseded by	None

Files	Date	Status	Document	Version
	2002-12-06	Standard	SC00084F.html, SC00084F.pdf	F
	2002-10-31	Experimental	XC00084E.pdf, XC00084E.html	E
	2001-08-15	Experimental	XC00084D.html, XC00084D.pdf	D
	2000-08-17	Experimental	XC00084C.pdf, XC00084C.html	C
	2000-11-30	Preliminary	PC00084B.pdf, PC00084B.html	B

Identifier	00075
Title	FIPA Agent Message Transport Protocol for IIOP Specification
Source	FIPA TC B
Type	Component
Subject	Transport Protocols
Summary	A description of a message transport protocol based on IIOP.
Keywords	FIPA,2000,Standards,Agents,Message Transport Service,MTS,Network Transport,Message Delivery,Network Protocols,IIOP
Superseded by	None

Files	Date	Status	Document	Version
	2002-12-06	Standard	SC00075G.html, SC00075G.pdf	G
	2002-10-31	Experimental	XC00075F.pdf, XC00075F.html	F
	2001-08-15	Experimental	XC00075E.html, XC00075E.pdf	E
	2000-08-17	Experimental	XC00075D.pdf, XC00075D.html	D
	2000-11-30	Experimental	XC00075C.pdf, XC00075C.html	C

Identifier	00071
Title	FIPA ACL Message Representation in XML Specification
Source	FIPA TC B
Type	Component
Subject	ACL Representations
Summary	A description of an ACL message representation in an XML encoding.
Keywords	FIPA,2000,Standards,Agents,Agent Communication Language,ACL,FIPA ACL,Message Representation,XML Encoding
Superseded by	None

Files	Date	Status	Document	Version

2002-12-06	Standard	SC00071E.html, SC00071E.pdf	E
2002-10-31	Experimental	XC00071D.pdf, XC00071D.html	D
2001-08-15	Experimental	XC00071C.html, XC00071C.pdf	C
2000-08-17	Experimental	XC00071B.pdf, XC00071B.html	B

Identifier	00070
Title	FIPA ACL Message Representation in String Specification
Source	FIPA TC B
Type	Component
Subject	ACL Representations
Summary	A description of an ACL message representation in a string encoding.
Keywords	FIPA,2000,Standards,Agents,Agent Communication Language,ACL,FIPA ACL,Message Representation,String Encoding
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00070I.html, SC00070I.pdf	I
	2002-10-31	Experimental	XC00070H.pdf, XC00070H.html	H
	2001-08-15	Experimental	XC00070G.html, XC00070G.pdf	G
	2000-12-01	Experimental	XC00070F.pdf, XC00070F.html	F
	2000-11-30	Experimental	XC00070E.pdf, XC00070E.html	E

Identifier	00069
Title	FIPA ACL Message Representation in Bit-Efficient Specification
Source	FIPA TC B
Type	Component
Subject	ACL Representations
Summary	A description of an ACL message representation in a bit-efficient encoding.
Keywords	FIPA,2000,Standards,Agents,Agent Communication Language,ACL,FIPA ACL,Message Representation,Bit-Efficient Encoding
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00069G.html, SC00069G.pdf	G
	2002-10-31	Experimental	XC00069F.pdf, XC00069F.html	F
	2001-08-15	Experimental	XC00069E.html, XC00069E.pdf	E
	2000-08-17	Experimental	XC00069C.pdf, XC00069C.html	C

Identifier	00067

Title	FIPA Agent Message Transport Service Specification
Source	FIPA TC B
Type	Component
Subject	Message Transport
Summary	A description of the Message Transport Service for agents on FIPA agent platforms.
Keywords	FIPA,2000,Standards,Agents,Message Transport Service,MTS,Network Transport,Message Delivery,Network Protocols
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00067F.html, SC00067F.pdf	F
	2002-10-30	Experimental	XC00067E.pdf, XC00067E.html	E
	2001-08-15	Experimental	XC00067D.html, XC00067D.pdf	D
	2000-08-17	Experimental	XC00067C.pdf, XC00067C.html	C

Identifier	00061
Title	FIPA ACL Message Structure Specification
Source	FIPA TC C
Type	Component
Subject	ACL
Summary	A description of the structure of FIPA ACL.
Keywords	FIPA,2000,Standards,Agents,Agent Communication Language,ACL,FIPA ACL,Structure
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00061G.html, SC00061G.pdf	G
	2002-10-31	Experimental	XC00061F.pdf, XC00061F.html	F
	2001-08-15	Experimental	XC00061E.html, XC00061E.pdf	E
	2000-08-17	Experimental	XC00061D.pdf, XC00061D.html	D

Identifier	00037
Title	FIPA Communicative Act Library Specification
Source	FIPA TC C
Type	Component
Subject	Communicative Acts
Summary	A library of FIPA communicative acts and requirements for new communicative acts.

Keywords	FIPA,2000,Standards,Agents,Library,Communicative Acts
Superseded by	None

Files	Date	Status	Document	Version
	2002-12-06	Standard	SC00037J.html, SC00037J.pdf	J
	2002-10-31	Experimental	XC00037I.pdf, XC00037I.html	I
	2001-08-15	Experimental	XC00037H.html, XC00037H.pdf	H
	2001-02-09	Experimental	XC00037G.pdf, XC00037G.html	G
	2000-08-17	Preliminary	PC00037F.pdf, PC00037F.html	F
	2000-11-30	Preliminary	PC00037E.pdf, PC00037E.html	E

Identifier	00036
Title	FIPA Propose Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Propose interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Propose
Superseded by	None

Files	Date	Status	Document	Version
	2002-12-06	Standard	SC00036H.html, SC00036H.pdf	H
	2002-10-31	Experimental	XC00036G.pdf, XC00036G.html	G
	2001-08-15	Experimental	XC00036F.html, XC00036F.pdf	F
	2001-02-09	Experimental	XC00036E.pdf, XC00036E.html	E
	2000-08-17	Preliminary	PC00036D.pdf, PC00036D.html	D

Identifier	00035
Title	FIPA Subscribe Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Subscribe interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Subscribe
Superseded by	None

Files	Date	Status	Document	Version
	2002-12-06	Standard	SC00035H.html, SC00035H.pdf	H
	2002-11-04	Experimental	XC00035G.html, XC00035G.pdf	G
	2001-08-15	Deprecated	DC00035F.html, DC00035F.pdf	F

2001-08-08	Deprecated	DC00035E.pdf, DC00035E.html	E
2000-08-17	Preliminary	PC00035D.pdf, PC00035D.html	D

Identifier	00034
Title	FIPA Recruiting Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Recruiting interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Recruiting
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00034H.html, SC00034H.pdf	H
	2002-10-31	Experimental	XC00034G.pdf, XC00034G.html	G
	2001-08-15	Experimental	XC00034F.html, XC00034F.pdf	F
	2001-02-09	Experimental	XC00034E.pdf, XC00034E.html	E
	2000-08-17	Preliminary	PC00034D.pdf, PC00034D.html	D

Identifier	00033
Title	FIPA Brokering Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Brokering interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Brokering
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00033H.html, SC00033H.pdf	H
	2002-10-31	Experimental	XC00033G.pdf, XC00033G.html	G
	2001-08-15	Experimental	XC00033F.html, XC00033F.pdf	F
	2001-02-09	Experimental	XC00033E.pdf, XC00033E.html	E
	2000-08-17	Preliminary	PC00033D.pdf, PC00033D.html	D

Identifier	00030
Title	FIPA Iterated Contract Net Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols

Summary	The FIPA Iterated Contract Net interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Iterated Contract Net
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00030H.html, SC00030H.pdf	H
	2002-10-31	Experimental	XC00030G.pdf, XC00030G.html	G
	2001-08-15	Experimental	XC00030F.html, XC00030F.pdf	F
	2001-02-09	Experimental	XC00030E.pdf, XC00030E.html	E
	2000-08-17	Preliminary	PC00030D.pdf, PC00030D.html	D

Identifier	00029
Title	FIPA Contract Net Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Contract Net interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Contract Net
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00029H.html, SC00029H.pdf	H
	2002-10-31	Experimental	XC00029G.pdf, XC00029G.html	G
	2001-08-15	Experimental	XC00029F.html, XC00029F.pdf	F
	2001-02-09	Experimental	XC00029E.pdf, XC00029E.html	E
	2000-08-17	Preliminary	PC00029D.pdf, PC00029D.html	D

Identifier	00028
Title	FIPA Request When Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Request When interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Request When
Superseded by	None

Files	Date	Status	Document	Version
	2002-12-06	Standard	SC00028H.html, SC00028H.pdf	H
	2002-10-31	Experimental	XC00028G.pdf, XC00028G.html	G
	2001-08-15	Experimental	XC00028F.html, XC00028F.pdf	F
	2001-02-09	Experimental	XC00028E.pdf, XC00028E.html	E
	2000-08-17	Preliminary	PC00028D.pdf, PC00028D.html	D

Identifier	00027
Title	FIPA Query Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Query interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Query
Superseded by	None

Files	Date	Status	Document	Version
	2002-12-06	Standard	SC00027H.html, SC00027H.pdf	H
	2002-10-31	Experimental	XC00027G.pdf, XC00027G.html	G
	2001-08-15	Experimental	XC00027F.html, XC00027F.pdf	F
	2001-02-09	Experimental	XC00027E.pdf, XC00027E.html	E
	2000-08-17	Preliminary	PC00027D.pdf, PC00027D.html	D

Identifier	00026
Title	FIPA Request Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Request interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Request
Superseded by	None

Files	Date	Status	Document	Version
	2002-12-06	Standard	SC00026H.html, SC00026H.pdf	H
	2002-10-31	Experimental	XC00026G.pdf, XC00026G.html	G
	2001-08-15	Experimental	XC00026F.html, XC00026F.pdf	F
	2001-02-09	Experimental	XC00026E.pdf, XC00026E.html	E
	2000-08-17	Preliminary	PC00026D.pdf, PC00026D.html	D

Identifier	00008

Title	FIPA SL Content Language Specification
Source	FIPA TC C
Type	Component
Subject	Content Languages
Summary	A description of the FIPA Semantic Language content language.
Keywords	FIPA,2000,Standards,Agents,Content Languages,Semantic Language,FIPA SL
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00008I.html, SC00008I.pdf	I
	2002-10-31	Experimental	XC00008H.pdf, XC00008H.html	H
	2001-08-15	Experimental	XC00008G.html, XC00008G.pdf	G
	2000-08-17	Experimental	XC00008F.pdf, XC00008F.html	F
	2000-11-30	Experimental	XC00008D.pdf, XC00008D.html	D

Identifier	00001
Title	FIPA Abstract Architecture Specification
Source	Architecture TC
Type	Component
Subject	Architecture
Summary	An abstract agent architecture for FIPA.
Keywords	FIPA,2000,Standards,Agents,Agent Architecture,Abstract Architecture
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-12-06	Standard	SC00001L.html, SC00001L.pdf	L
	2002-10-31	Experimental	XC00001K.pdf, XC00001K.html	K
	2001-08-15	Experimental	XC00001J.html, XC00001J.pdf	J
	2001-02-09	Experimental	XC00001I.pdf, XC00001I.html	I
	2001-01-02	Preliminary	PC00001H.pdf, PC00001H.html	H
	2000-11-30	Preliminary	PC00001D.pdf, PC00001D.html	D
	2000-11-30	Preliminary	PC00001C.pdf, PC00001C.html	C

Bijlage III: Experimental en Preliminary specificaties van FIPA

(http://www.fipa.org/specs/index.html)

Identifier	00095			
Title	FIPA Agent Discovery Service Specification			
Source	FIPA TC Ad Hoc			
Type	Component			
Subject	Management			
Summary	A description of agent discovery in Ad Hoc networks			
Keywords	FIPA,Standards,Agents,Management,Discovery,Ad Hoc Network			
Superseded by	None			
Files	**Date**	**Status**	**Document**	**Version**
	2003-11-10	Preliminary	PC00095A.pdf	A
	2003-10-20	Preliminary	PC00095.pdf	Initial

Identifier	00096			
Title	FIPA JXTA Discovery Middleware Specification			
Source	FIPA TC Ad Hoc			
Type	Component			
Subject	Management			
Summary	A description of agent discovery in Ad Hoc networks using JXTA			
Keywords	FIPA,Standards,Agents,Management,Discovery,Ad Hoc Network,JXTA			
Superseded by	None			
Files	**Date**	**Status**	**Document**	**Version**
	2003-11-10	Preliminary	PC00096A.pdf	A
	2003-10-20	Preliminary	PC00096.pdf	Initial

Identifier	00010
Title	FIPA KIF Content Language Specification
Source	FIPA TC C
Type	Component
Subject	Content Languages
Summary	A description of a FIPA content language based on the Knowledge Interchange Format.

Keywords	FIPA,2000,Standards,Agents,Content Languages,Knowledge Interchange Format,KIF,FIPA KIF
Superseded by	None

Files	Date	Status	Document	Version
	2003-01-29	Experimental	XC00010C.pdf, XC00010C.html	C
	2001-08-15	Experimental	XC00010B.html, XC00010B.pdf	B
	2000-08-17	Experimental	XC00010A.pdf, XC00010A.html	A

Identifier	00089
Title	FIPA Domains and Policies Specification
Source	Architecture TC
Type	Component
Subject	Architecture
Summary	A description of domains and policies for the FIPA Abstract Architecture.
Keywords	FIPA,2000,Standards,Agents,Abstract Architecture,Domains,Policies
Superseded by	None

Files	Date	Status	Document	Version
	2001-08-15	Preliminary	PC00089D.html, PC00089D.pdf	D
	2001-02-16	Preliminary	PC00089C.pdf, PC00089C.html	C
	2001-01-05	Preliminary	PC00089B.pdf, PC00089B.html	B

Identifier	00093
Title	FIPA Messaging Interoperability Service Specification
Source	Gateways TC
Type	Component
Subject	Message Transport
Summary	A service for interoperability between different message formats
Keywords	FIPA,2000,Standards,Agents,MTP Translation,Message Envelope Encoding Translation,ACL Encoding Translation,CL Encoding Translation,Messaging Interoperability,Gateways
Superseded by	None

Files	Date	Status	Document	Version
	2002-04-22	Experimental	XC00093B.html, XC00093B.pdf	B
	2001-08-15	Preliminary	PC00093A.html, PC00093A.pdf	A
	2001-08-07	Preliminary	PC00093.html	Initial

Identifier	00092

Title	FIPA Message Buffering Service Specification
Source	Gateways TC
Type	Component
Subject	Applications
Summary	A message buffering service for the FIPA Message Transport Service for wireless or roaming devices
Keywords	FIPA,2000,Standards,Agents,Gateways,Message Buffering,Roaming
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2002-04-22	Experimental	XC00092B.html, XC00092B.pdf	B
	2001-08-15	Preliminary	PC00092A.html, PC00092A.pdf	A
	2001-08-07	Preliminary	PC00092.html	Initial

Identifier	00086
Title	FIPA Ontology Service Specification
Source	FAB
Type	Component
Subject	ACL
Summary	The description of an Ontology Agent for registering and serving ontologies to agents.
Keywords	FIPA,2000,Standards,Agents,Agent Communication,Ontologies,Ontolingua
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2001-08-15	Experimental	XC00086D.html, XC00086D.pdf	D
	2000-08-17	Experimental	XC00086C.pdf, XC00086C.html	C

Identifier	00083
Title	FIPA Personal Assistant Specification
Source	FAB
Type	Informative
Subject	Applications
Summary	A description of agents and an ontology for personal assistant applications.
Keywords	FIPA,2000,Standards,Agents,Applications,Personal Assistant
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**

2001-08-15	Experimental	XC00083B.html, XC00083B.pdf	B
2000-10-17	Experimental	XC00083A.pdf, XC00083A.html	A
2000-11-30	Preliminary	PC00083.pdf, PC00083.html	Initial

Identifier	00082
Title	FIPA Network Management and Provisioning Specification
Source	FAB
Type	Informative
Subject	Applications
Summary	A description of agents and an ontology for network management and provisioning applications.
Keywords	FIPA,2000,Standards,Agents,Applications,Network Management,Network Provisioning
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2001-08-15	Experimental	XC00082B.html, XC00082B.pdf	B
	2000-10-17	Experimental	XC00082A.pdf, XC00082A.html	A
	2000-11-30	Preliminary	PC00082.pdf, PC00082.html	Initial

Identifier	00081
Title	FIPA Audio-Visual Entertainment and Broadcasting Specification
Source	FAB
Type	Informative
Subject	Applications
Summary	A description of agents and an ontology for supporting entertainment and broadcasting applications.
Keywords	FIPA,2000,Standards,Agents,Applications,Audio Visual,Entertainment,Broadcasting,TV Programming,Films,Movies,Video Games
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2001-08-15	Experimental	XC00081B.html, XC00081B.pdf	B
	2000-10-17	Experimental	XC00081A.pdf, XC00081A.html	A
	2000-11-30	Preliminary	PC00081.pdf, PC00081.html	Initial

Identifier	00080
Title	FIPA Personal Travel Assistance Specification

Source	FAB
Type	Informative
Subject	Applications
Summary	A description of agents and an ontology for supporting personal travel assistance applications.
Keywords	FIPA,2000,Standards,Agents,Applications,Personal Travel Assistance,Flight Reservations,Hotel Reservations
Superseded by	None

Files	Date	Status	Document	Version
	2001-08-15	Experimental	XC00080B.html, XC00080B.pdf	B
	2000-10-17	Experimental	XC00080A.pdf, XC00080A.html	A
	2000-11-30	Preliminary	PC00080.pdf, PC00080.html	Initial

Identifier	00079
Title	FIPA Agent Software Integration Specification
Source	FAB
Type	Component
Subject	Applications
Summary	A description of agents and an ontology for supporting agent integration with software systems.
Keywords	FIPA,2000,Standards,Agents,Software Integration,Wrappers,Legacy Systems
Superseded by	None

Files	Date	Status	Document	Version
	2001-08-15	Experimental	XC00079B.html, XC00079B.pdf	B
	2000-08-17	Experimental	XC00079A.pdf, XC00079A.html	A

Identifier	00076
Title	FIPA Agent Message Transport Protocol for WAP Specification
Source	FIPA TC E
Type	Component
Subject	Transport Protocols
Summary	A description of a message transport protocol based on WAP.
Keywords	FIPA,2000,Standards,Agents,Message Transport Service,MTS,Network Transport,Message Delivery,Network Protocols,WAP
Superseded by	None

Files	Date	Status	Document	Version

2001-08-15	Experimental	XC00076C.html, XC00076C.pdf	C
2000-08-17	Experimental	XC00076B.pdf, XC00076B.html	B

Identifier	00032
Title	FIPA Dutch Auction Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA Dutch Auction interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,Dutch Auction
Superseded by	None

Files	Date	Status	Document	Version
	2001-08-15	Experimental	XC00032F.html, XC00032F.pdf	F
	2001-02-09	Experimental	XC00032E.pdf, XC00032E.html	E
	2000-08-17	Preliminary	PC00032D.pdf, PC00032D.html	D

Identifier	00031
Title	FIPA English Auction Interaction Protocol Specification
Source	FIPA TC C
Type	Component
Subject	Interaction Protocols
Summary	The FIPA English Auction interaction protocol.
Keywords	FIPA,2000,Standards,Agents,Interaction Protocols,Conversations,English Auction
Superseded by	None

Files	Date	Status	Document	Version
	2001-08-15	Experimental	XC00031F.html, XC00031F.pdf	F
	2001-02-09	Experimental	XC00031E.pdf, XC00031E.html	E
	2000-08-17	Preliminary	PC00031D.pdf, PC00031D.html	D

Identifier	00011
Title	FIPA RDF Content Language Specification
Source	FIPA TC C
Type	Component
Subject	Content Languages
Summary	A description of a FIPA content language based on the Resource Description

	Framework.
Keywords	FIPA,2000,Standards,Agents,Content Languages,Resource Definition Format,RDF,FIPA RDF
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2001-08-15	Experimental	XC00011B.html, XC00011B.pdf	B
	2000-08-17	Experimental	XC00011A.pdf, XC00011A.html	A

Identifier	00009
Title	FIPA CCL Content Language Specification
Source	FIPA TC C
Type	Component
Subject	Content Languages
Summary	A description of the FIPA Constraint Choice Language content language.
Keywords	FIPA,2000,Standards,Agents,Content Languages,Semantic Language,FIPA SL
Superseded by	None

Files	**Date**	**Status**	**Document**	**Version**
	2001-08-15	Experimental	XC00009B.html, XC00009B.pdf	B
	2000-08-17	Experimental	XC00009A.pdf, XC00009A.html	A

Bijlage IV: Randinformatie SKF

(informatie met de heer M. Linten)

Inleiding

De SKF groep is een wereldwijde producent van vooral lagers maar ook andere producten zoals dichtingen en smeerproducten. Naast deze producten bieden ze ook diensten en oplossingen voor de industrie aan in de vorm van technische ondersteuning, onderhoud, monitoring en opleidingen. SKF heeft wereldwijd 100 productiefaciliteiten met verkoopssites in een 70-tal landen. SKF handelt zelf zijn logistieke processen af en wordt daarbij ondersteund door een 15000-tal distributeurs en dealers wereldwijd. (SKF.com, 2007)

Om een goed begrip van deze concrete case te krijgen is het belangrijk dat de logistieke Europese structuur van SKF uit de doeken gedaan wordt. Tot 1994 was het logistiek proces zo opgebouwd dat heel Europa beleverd werd vanuit de productiefaciliteiten die ook fungeerden als magazijn. In 1994 veranderde het bedrijfsmodel op logistiek gebied in die zin dat er een Europees Distributiecentrum (EDC) opgericht werd in Tongeren. De stockage en distributie van de productiefaciliteiten uit de Europese landen gebeurt sinds toen in en vanuit Tongeren.

Naast het EDC zijn er nog vier zogenoemde internationale hubs opgericht in 1994, namelijk in Tours, Raskar, Schweinfurt en Göteborg. In de landen waar deze steden liggen zijn tevens belangrijke productiefaciliteiten. Het spreekt voor zich dat er tussen de hubs en het EDC te Tongeren frequente transportritten zijn. Enerzijds worden hubs beleverd vanuit Tongeren en anderzijds dienen de materialen uit de fabrieken in deze landen aangeleverd te worden te Tongeren om deze goederen over Europa te kunnen verdelen. Naast deze taak staan de internationale hubs ook in voor directe belevering voor klanten in dat specifieke land.

Aangezien de klanten van SKF zich ook buiten deze landen bevinden, zijn er in andere landen zogenoemde kleinere ‘Domestic Hubs’. Deze domestic hubs die bevoorraad worden door de internationale hubs of het EDC verzorgen specifiek de belevering van de

klanten. Het beheer van de domestic hubs is voor de meeste gevallen in de handen van SKF zelf maar wordt voor de rest uitbesteed aan derden.

Het Daily Transport Schedule (DTS), dit is het transport tussen het EDC en een internationale hub, twee internationale hubs onderling of een internationale hub en een domestic hub, wordt centraal gemonitord vanuit Zweden. Deze transportroutes worden ook wel 'line-hauls' genoemd. Voor het 'Domestic Transport', dit is het transport tussen een domestic hub en de klant, is een aparte verantwoordelijke per regio aangewezen. Weliswaar krijgt de verantwoordelijke centrale dienst te Zweden de nodige ondersteuning van de regionale verantwoordelijken. Zo zal een regionale verantwoordelijke informatie meedelen over de transporteur en meldingen doen over de line-haul die verbonden is met zijn hub of EDC.

Een ander belangrijk punt in de logistieke organisatie van SKF is dat zij voor deze uitgebreide en volumineuze transportbehoeftes volledig terugvallen op de transportdiensten van derden. Het is dus zo dat SKF zelf geen enkele vrachtwagen bezit of chauffeur in dienst heeft. Tot slot kunnen we nog vermelden dat SKF via zijn uitstekend logistiek netwerk ook de distributie van externe partijen voor hun rekening neemt. Het is immers zo dat de producten van SKF (vooral lagers) een klein volume innemen zodat een geladen vrachtwagen nog veel leeg volume heeft. Externe partijen die in aanmerking komen, hebben dan ook vooral grote en lichte producten die dienen gedistribueerd te worden. Op deze manier wordt er een win-win situatie gecreëerd daar SKF de kosten kan delen en de externe partij niet zelf deze distributie moet organiseren en de infrastructuur moet voorzien.

Relatie met logistieke dienstverleners

De relatie tussen de transporteurs en SKF is gebaseerd op een lange termijn relatie. Elke line-haul wordt uitgebaat door één transporteur. Bij het domestic transport (vanuit de domestic hubs naar de klanten) is deze situatie enigszins anders aangezien dit transport naar de klant wordt gekenmerkt door een iets complexer en dynamischer karakter. Het gaat hier meestal over een kleinere laadeenheid die vaak specifieke kenmerken heeft zoals de manier waarop de goederen gestockeerd zijn (paletten, containers…) die aan de

klanten geleverd dienen te worden. Daarom is het niet mogelijk met één enkele leverancier te werken en wordt er dus beroep gedaan op meerdere transporteurs.

De verantwoordelijke voor het DTS te Zweden staat in voor de bepaling van het vaste volume dat dagelijks getransporteerd dient te worden op een bepaalde line-haul. Op deze manier zorgt hij dus voor een zekere stabiliteit in de goederenstroom van SKF. Dit gebeurt via de registratie van de fill-rates en mogelijke afwijkingen van de vastgestelde capaciteit. Op deze manier komt men tot een geschat volume dat tussen twee punten dient getransporteerd te worden. SKF zal dan een aantal vrachtwagens vast boeken zodat het verwacht dagelijks volume getransporteerd kan worden. Deze verantwoordelijke staat ook in voor evaluatie van de transporteurs (performance reporting) van de transporteurs die de line-haul voor hun rekening nemen.

De logistieke organisatie van SKF speelt naast het vastgeboekte deel van het transport ook in op een flexibele invulling van de transportbehoefte. Het dagelijks volume per 'line-haul' wordt immers door transportplanners in de 'hubs' en het EDC gepland en opgevolgd. SKF kan dan een optie lichten om de vastgeboekte vrachtwagens te annuleren. In de contracten wordt ook de mogelijkheid gestipuleerd om vrachtwagens bij te boeken tegen een bepaald tijdstip. Indien men vaak in problemen komt met de gereserveerde capaciteit zullen de verantwoordelijke voor het DTS en de planners overgaan tot het boeken van een extra vaste transportrit of net het schrappen van een transportrit. Het is dus zo dat de gereserveerde capaciteit zo goed als mogelijk afgestemd wordt op de transportbehoefte. De kunst bij SKF is dus het evenwicht zoeken tussen vast boeken en 'last minute' boeken. Het is evident dat deze laatste mogelijkheid iets duurder is en niet in alle situaties gegarandeerd kan worden.

Verder kunnen we stellen dat het systeem van SKF klokvast is. Per 'line-haul' worden vaste vertrek- en aankomsttijdstippen voor de vrachtwagens van de transporteurs in het contract vermeld. Ook uiterste tijdstippen voor het reserveren van transportritten worden hierin gespecificeerd. Deze klokvastheid is naast de planning voor SKF ook een bron van stabiliteit voor de planning van transporteurs. Zij hebben een vrij duidelijk beeld van de bezette vrachtwagens en weten wanneer deze vrachtwagens operationeel moeten zijn. Enkel het omgaan met de variabele transportritten is een onduidelijkheid.

De criteria om met een transporteur in zee te gaan zijn in dalende volgorde van belangrijkheid: prijs, betrouwbaarheid, data-integratie met SKF en milieuvriendelijkheid van transportmiddelen. Daarnaast vereist SKF dat deze transporteurs aan 'performance reporting' doen. We zien dus dat SKF over de macht bezit om bepaalde eisen op te leggen aan hun transportleveranciers. Zo wordt er bij SKF toch een 96 à 97% leverbetrouwbaarheid als norm gesteld. Eisen die zeker zo belangrijk zijn als deze leverbetrouwbaarheid is de mate van data-integratie tussen de leverancier en SKF.

De integratie van data tussen transporteur en SKF wordt geregeld via een centrale EDI-server van SKF. Aan de hand van deze server gebeurt ook de controle en opvolging van het DTS te Zweden. Tijdens het interview liet Mr. Linten duidelijk blijken dat het implementeren van deze technologie bij transporteurs kritisch is voor het afsluiten van het contract. In feite komt het er op neer dat als er geen EDI communicatie kan tot stand gebracht worden ook niet met de transporteur in zee gegaan wordt. De communicatie van orders en vrachtdocumenten verlopen allemaal via EDI berichtuitwisseling naar de centrale server. Op deze manier ondersteunt EDI de Global Tracking Service van SKF waardoor men op elk moment de status en positie van een pakje kan terugvinden. Deze EDI berichtuitwisseling zorgt er dus voor dat de reisweg van pakjes accuraat opgevolgd kan worden. De informatie over deze pakjes wordt centraal gemonitord te Zweden via de informatie die terecht komt op de EDI server. Dit team controleert dus of de integratie van verschillende transportritten foutloos gebeurt.

We kunnen stellen dat de kenmerken van EDI die reeds besproken zijn in hoofdstuk 2 overeenkomen met de uitlatingen van de heer Linten. Zo realiseert hij zich dat SKF de transporteurs dwingt tot kostelijke implementaties. Toch is dit logisch vermits SKF hen dergelijk groot volume kan garanderen en stabiliteit in de planning van hun bedrijfsmiddelen verzekert. Een nadeel van EDI is dat het overschakelen tussen leveranciers bemoeilijkt wordt. Door de kostelijke en tijdrovende implementatie is het niet evident om op een korte termijn een nieuwe transporteur te vinden voor de uitbating van een bepaalde route. Tot slot kan men stellen dat EDI kritisch is voor SKF daar het logistieke processen in de keten op elkaar afstemt via de centrale server. Wanneer EDI dus niet functioneert of bepaalde problemen ontstaan bij bestand uitwisseling, is het moeilijk om deze informatie op een andere manier in het systeem te 'forceren' zodat toch de oorspronkelijke planning uitgevoerd kan worden. De afhankelijkheid van EDI samen met de klokvastheid van het systeem zorgen ervoor dat een onzekerheid of onvoorziene

omstandigheden in een transportketen zich zal verspreiden door heel de transportketen en moeilijk te corrigeren valt in een later stadium.

Tot slot kunnen we vermelden welke aspecten overeengekomen worden door SKF en de transporteur in een contract. Het contract bezit algemene voorwaarden (aard van overeenkomst, contactpersonen, start en einde van het contract …) en als tweede deel 'Standard Operational Procedures' (SOPs). Hier wordt onder andere de informatiestroom beschreven en de fysische prestaties en prijzen voor vaste en variabele trucks vastgelegd. Al deze voorwaarden worden ook centraal beheerd in een database te Zweden. Deze informatie kan immers nuttig zijn als opzoekmateriaal voor de kostprijs van een volledig traject of als maatstaf voor de prijzen van nieuwe transporteurs.

Bijlage V: Use case template

Basic Use Case Template

Alistair Cockburn
email: arc@acm.org
home page: http://alistair.cockburn.us

Document: TR.96.03a
This Version Date: October 26, 1998
Version: 2
Previous Version Date: April 26, 1996

USE CASE #	< the name is the goal as a short active verb phrase>	
Goal in Context	<a longer statement of the goal in context if needed>	
Scope & Level	<what system is being considered black box under design> <one of : Summary, Primary Task, Subfunction>	
Preconditions	<what we expect is already the state of the world>	
Success End Condition	<the state of the world upon successful completion>	
Failed End Condition	<the state of the world if goal abandoned>	
Primary, Secondary Actors	<a role name or description for the primary actor>. <other systems relied upon to accomplish use case>	
Trigger	<the action upon the system that starts the use case>	
DESCRIPTION	**Step**	**Action**
	1	<put here the steps of the scenario from trigger to goal delivery,and any cleanup afte>
	2	<...>
	3	
EXTENSIONS	**Step**	**Branching Action**
	1a	<condition causing branching> : <action or name of sub.use case>
SUB-VARIATIONS		**Branching Action**
	1	<list of variation s>

Auteursrechterlijke overeenkomst

Opdat de Universiteit Hasselt uw eindverhandeling wereldwijd kan reproduceren, vertalen en distribueren is uw akkoord voor deze overeenkomst noodzakelijk. Gelieve de tijd te nemen om deze overeenkomst door te nemen, de gevraagde informatie in te vullen (en de overeenkomst te ondertekenen en af te geven).

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Een kritische analyse van agenttechnologie als ondersteuning voor logistieke netwerken
Richting: **Handelsingenieur in de beleidsinformatica** Jaar: **2007**
in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Ik ga akkoord,

Mario AERTS

Datum: **03.06.2007**