De totale samenhang tussen de diagramsoorten in UML

Peter Bollen
Departement Management Wetenschappen
Universiteit Maastricht
December 2002

Samenvatting. In dit artikel worden de 9 UML diagramsoorten in hun onderlinge samenhang beschreven. Als uitgangspunt voor deze beschrijving worden de modelleringsconstructies en de modelsoorten uit Kenniskunde gebruikt. Het artikel laat verder zien wat de essentiële diagram-soorten in UML zijn en in welke volgorde deze UML diagrammen dienen te worden gemaakt in een praktijksituatie.

1. Inleiding

De laatste jaren is in de wereld van de conceptuele bedrijfsmodellering (dat is het opstellen van een regelgeving voor een administratie ter ondersteuning van de bedrijfsprocessen) een trend zichtbaar naar het gebruik van objekt-geöriënterende modelleringstalen. Met name de door het Amerikaanse IT-bedrijf Rational ontworpen en door de OMG (de Objekt Management Group, een organisatie waarin vertegenwoordigers van de ICT industrie en de overheid deelnemen, die samen komen tot IT-standaarden) geaccepteerde modelleringstaal Unified Modeling Language (UML) is uitgegroeid tot een de-facto modelleringstandaard in de Amerikaanse IT-industrie en wordt ondersteund in de nieuwe *Microsoft Visual studio.net* ® ontwikkelomgeving [9, 10]. In dit artikel zullen we nagaan in hoeverre UML geschikt is voor het maken van conceptuele bedrijfsmodellen, die bijvoorbeeld gebruikt kunnen worden voor het presenteren van de resultaten uit de analysefase van een IT-project. Als 'benchmark' voor UML zullen we een conceptuele bedrijfsmodelleringmethodiek gebruiken die met name in Nederland een redelijk grote bekendheid geniet en die onderdeel is van *Kenniskunde*. Een goede inleiding in Kenniskunde is te vinden in [11].

In sectie 2 zullen we een beknopt overzicht geven van de modelsoorten in Kenniskunde die de informatiekundige aspecten van een bedrijfsproces-ondersteuningssysteem volledig beschrijven. In sectie 3 zullen we een overzicht geven van de modelsoorten en perspektieven die in UML worden gebruikt. In sectie 4 zullen we aan de hand van een voorbeeld het gebruik van de modelsoorten in het kader van de conceptuele bedrijfsmodellering behandelen en illustreren hoe de UML diagramsoorten samenhangen en welke onderdelen van de Kenniskunde modellen gerepresenteerd kunnen worden in de UML diagramsoorten. In sectie 4 laten we zien hoe de UML diagramsoorten gebruikt kunnen worden om het *informatie-, proces-* en het *gedrags*perspektief van een bedrijfssysteem te modelleren. In sectie 5 tenslotte worden conclusies gegeven.

2. Kenniskunde

Kenniskunde is onder andere een modelleringmethodiek die is voortgekomen uit de informatie-analyse methoden *NIAM* [13, 18] en *Universele Informatiekunde* [2, 12, 15, 16]. Het aantal modelleringconstructies in Kenniskunde is echter kleiner en de informatiediagrammen zijn compacter ten opzichte van de voorgangers. Een verdere verbetering t.o.v. NIAM en Universele Informatiekunde in Kenniskunde is de eenduidige beschrijving van de betekenis van de begrippen. Het geeft alle beperkingsregels en geeft aan hoe de inhoud van de administratie gecommuniceerd moet worden. Een andere belangrijke uitbreiding van Kenniskunde t.o.v. NIAM en Universele Informatiekunde is de uitbreiding van de modelleringsconstructies voor het *informatieperspektief* met modelleringsconstructies voor het *procesperspektief* en het *gedragsperspektief.*

2.1. Casus auto-verhuur

Het voorbeeld dat we gebruiken in dit artikel is het voorbeeld dat beschreven staat in de beschrijving van microsoft Visual Studio.Net (te vinden op http: //msdn.microsoft.com/ vstudion/ nextgen/ technology/modelsoftware.asp bezocht op 22 augustus 2002). Het betreft hier voorbeeld dat bestaat uit een *klantoverzicht*, een *vlootoverzicht*, een *personeelsoverzicht*, een *filialenoverzicht* en een *huurovereenkomst* van een autoverhuurbedrijf. Deze concrete gebruikersvoorbeelden worden ook wel *data use-cases* genoemd [8]. Een significant voorbeeld binnen het universum van discussie (UvD) dat hoort tot de autoverhuur casus is de data use-case *klantoverzicht* uit figuur 2. Een overzicht van de definities van de gebruikte begrippen in dit voorbeeld wordt gegeven in figuur 1. Herst heeft een zestigtal filialen in Nederland. Een *filiaal* kan wordt aangeduid door middel van een *filiaal ID*. Een klant wordt aangeduid door middel van een *klantnummer*. Elke klant van Herst wordt opgenomen in het reserveringssysteem van Herst autoverhuur. Van elke *klant* wordt eveneens de *klantnaam*

Begrip	**Definitie van het Begrip**
Klant	Een persoon die ooit een auto bij Herst heeft geleend of op het punt staat om een auto bij Herst te lenen.
Datum	Aanduiding voor een specifieke dag in het verleden, heden of toekomst
Rijbewijs	Een verklaring van de Nederlandse staat dat de persoon die wordt vermeld in het rijbewijs een auto, motor en/of vrachtwagen mag besturen, binnen de termijn die in het rijbewijs staat vermeld.
Adres	Een aanduiding van de woonplaats van een persoon of de vestigingsplaats van een vestiging van Herst autoverhuur.
Telefoonaansluiting	Een vaste of mobiele telefoonverbinding.
Postgebied	Een door de Nederlandse posterijen gedefinieerde verzameling adressen.

Fig. 1: Definities van begrippen uit het UvD van figuur 2

vastgelegd. Verder wordt er van elke klant het *adres, postgebied* en *telefoonaansluiting* vastgelegd. Een *adres* wordt aangeduid door een *adrescode* die bestaat uit een combinatie van *straatnaam, huisnummer*, en *woonplaatsnaam*. Een *postgebied* wordt aangeduid door een *postcode*. Een *telefoonaansluiting* wordt aangeduid door een *telefoonnummer*. Er wordt van een klant eveneens de *geboortedatum* opgeslagen. Een geboortedatum kan worden aangeduid als *datum* met een *datumcode* dd-mm-jjjj, bestaande uit een dagcode, een maandcode en een jaartal. Er wordt tevens een rijbewijs aangeduid door een rijbewijscode vastgelegd tezamen met een aanduiding van minimaal één rijbewijstype voor elke klant. Een *rijbewijstype* moet één van de volgende waarden hebben: *BE*, *CE* of *DE*. Tenslotte wordt voor elke klant een *verhuurverleden* bijgehouden. Bij het inschrijven van een klant wordt dit automatisch gelijkgesteld aan *goed*. Indien echter op een verhuurovereenkomst van een klant op enig moment éénmaal de terugbrengstatus *te laat* voorkomt dan wordt het verhuurverleden gelijkgesteld aan *fout*. In zo'n geval wordt er door de verhuuragent besloten om eventueel een borgsom te eisen of om de klant om additionele identificatie documenten te vragen, voordat hij/zij een nieuwe verhuurovereenkomst kan afsluiten (zie ook de beschrijving van de huurovereenkomst in figuur 6).

HERST AUTOVERHUUR KLANTOVERZICHT

Klantnummer: *0078* **Klantnaam:** *Willemsen*
Adres: *steenstraat 5, Zoetermeer* **postgebied**: *4567 hj,*

Telefoon: *0031-04567-45678*
Geboortedatum: *12-11-1970*
Code van Rijbewijs: *hjk-89670-vbngh* **Type Rijbewijs**: *BE*
Verhuurverleden: *goed*

Fig. 2: voorbeeld (data use case) klantoverzicht

Verder wordt er in elke filiaal van Herst een vlootoverzicht bijgehouden waarop de op dat moment in dienstzijnde leenautos staan vermeld. Een *filiaal* wordt aangeduid door een *filiaal ID*. Een *leenauto* wordt aangeduid binnen Herst door een *leenautonummer*. Van elke *leenauto* worden de volgende zaken vastgelegd: het *model*, het *kenteken*, een indicatie van het aantal kilometers tot de volgende onderhoudsbeurt (aangeduid als *onderhoudstatus*) en de *verhuurstatus*. Een onderhoudstatus wordt uitgedrukt als een *geheel getal*. Verder wordt voor een specifiek *model* in een specifiek *filiaal* een *dagtarief* vastgesteld. Dit betekent dat het huren van een leenauto van hetzelfde model in sommige filialen goedkoper kan zijn dan in andere filialen. Een *model* wordt geïdentificeerd door een *modelnaam*. Het *dagtarief* wordt uitgedrukt als een *bedrag* in Euro's uitgedrukt als een geheel getal. Een *kenteken* bestaat uit 3 combinaties van 2 karakters. Elk van deze 3 combinaties bestaat uitsluitend uit oftewel letters of cijfers. De *verhuurstatus* van een auto geeft aan of een auto is uitgeleend (verhuurstatus=uitgeleend), een auto in onderhoud is (verhuurstatus=in onderhoud) of dat een auto beschikbaar is om uitgeleend te worden (verhuurstatus=op voorraad).

HERST AUTOVERHUUR VLOOTOVERZICHT FILIAAL: *56*

Leenauto Nummer	Model	Kenteken	Onderhoud status	Verhuur status	Dagtarief
1	*Opel Astra*	*45-gh-23*	*12.500*	*Uitgeleend*	*€ 75*
7	*Renault 88*	*hj-45-78*	*15.000*	*Op Voorraad*	*€ 80*
16	*Ford Kia*	*67-34-kl*	*0*	*In onderhoud*	*€ 90*
17	*Ford Kia*	*ce-34-62*	*0*	*Op voorraad*	*€ 90*

Fig. 3: voorbeeld (data use case) vlootoverzicht

Elke filiaal heeft een of meer *verhuuragenten*. Sommige filialen hebben tevens een onderhoudsfaciliteit waar *monteurs* werken die in dienst zijn van Herst autoverhuur. Verhuuragenten en monteurs vormen de werknemers van Herst autoverhuur. Een *werknemer* wordt binnen Herst Nederland aangeduid door een *werknemercode*. Van elke werknemer wordt de *werknemernaam*, het *filiaal* waar hij/zij werkzaam is, het *adres*, het *postgebied*, de *telefoonaansluiting* die men thuis heeft en de *datum van indiensttreding* vastgelegd. Van elke verhuuragent wordt tevens vastgelegd wat zijn/haar *commissie* is uitgedrukt in een commissiepercentage van de gemaakte verkopen is en wordt er vastgelegd wat de jaarlijkse verkoopdoelstelling is. Een *verkoopdoelstelling* wordt uitgedrukt als *bedrag*. Van een monteur wordt vastgelegd wat zijn/haar *specialisatie* is aangeduid door een *specialisatiecode* en welke *opleiding* (uitgedrukt door een *opleidingscode*) hij/zij heeft genoten.

HERST AUTOVERHUUR PERSONEELS OVERZICHT FILIAAL: *56*

VERHUURAGENTEN

werknemer Code	Naam	Adres	Postgebied	Tel.	Datum in dienst	Commissie percentage	Verkoop doelstelling
1007	*Pietersen*	*Markt 25*	*Schulle 6734 jk*	*045-4567829*	*12-10-1988*	*15*	*€ 100 000,-*
1222	*Willemsen*	*Steenstr. 34*	*Hens 5679 kl*	*047-7815267*	*03-05-1993*	*10*	*€ 200 000,-*

MONTEURS

Code	Naam	Adres	Postgebied	Tel.	Datum in dienst	Opleiding	Specialisatie
1008	*Evers*	*Plein 5*	*Schulle 6735 jk*	*045-4566729*	*12-10-1988*	*LTS-C*	*Mechanisch*
1019	*Jans*	*Stokstr. 4*	*Sibbe 6783 aa*	*007-9467367*	*03-05-1993*	*MTS*	*Carosserie*

Fig. 4: voorbeeld (data use case) personeelsoverzicht

Herst Nederland gebruikt eveneens een overzicht van filialen waarbij voor elke filiaal het adres wordt vermeld, het *postgebied* waarin het filiaal zich bevindt, de *telefoonaansluiting* van het filiaal en de *werknemercode* en de *naam* en werknemercode van de *filiaalmanager*. De filiaalmanager is altijd een verhuuragent van diezelfde vestiging.

HERST AUTOVERHUUR FILIALENOVERZICHT

Filiaal	Adres	Postgebied	Telefoonaansluiting	Manager Werknemercode Naam
56	*Vliethoof 12 Vecht*	*2378 AB*	*04568-3546728*	*1007 Pietersen*
57	*Dam 2 Amsterdam*	*4578 AG*	*04568-3546728*	*1119 Schaafsma*

Fig. 5: voorbeeld (data use case) filialenoverzicht

Een ander document binnen Herst autoverhuur is de *huurovereenkomst* (zie figuur 6). Een *huurovereenkomst* wordt aangeduid door een *huurovereenkomstcode*. In een huurovereenkomst wordt allereerst vastgelegd voor welke klant een reservering wordt gedaan en op welke dag dit plaatsvindt. Verder worden er bij de reservering de volgende klantgegevens uit het Herst informatiesysteem opgenomen in de huurovereenkomst: *klantnaam*, *adres, postgebied, telefoonaansluiting, rijbewijs*, *rijbewijstype*. Verder wordt vastgelegd welk model auto de klant nodig heeft en gedurende welke periode. Zo'n periode is in principe een tijds-interval dat begint met de reserveringsstartdatum en eindigt met de reserveringseinddatum [1]. Op het moment dat de klant de auto op komt halen wordt het kenteken van de leenauto die daadwerkelijk wordt uitgeleend vastgelegd. Bij het terugbrengen van de leenauto wordt het af te rekenen bedrag berekend en kan worden geregistreerd of de auto al dan niet te laat is teruggebracht door het toekennen van een waarde voor de terugbrengstatus. In het geval de auto op tijd wordt teruggebracht wordt dit de waarde *op tijd*. In het geval de vooraf vastgestelde uiterste terugbrengdatum wordt overschreden krijgt de terugbrengstatus de waarde *te laat.* Alleen in het geval dat er vooraf *geen* uiterste verhuurdatum is overeengekomen krijgt de terugbrengstatus na het terugbrengen de waarde *auto is ingeleverd*. Verder wordt de inhoud van de tank bij terugkomst vastgesteld. Deze dient oftewel vol of leeg te zijn. Indien de tank noch vol noch leeg is wordt er leeg ingevuld. Bij het terugbrengen wordt tevens de kilometerstand door een werknemer van Herst vastgelegd. Het is mogelijk dat een klant een auto terugbrengt bij een ander filiaal van Herst dan het filiaal dat de auto heeft uitgeleend. In zo'n geval wordt er expliciet vastgelegd bij welk filiaal de auto wordt teruggebracht: het *transitfiliaal*. Een transitfiliaal is *altijd* een ander filiaal, dan de filiaal waar de huurovereenkomst is afgesloten.

HERST AUTOVERHUUR HUUROVEREENKOMST *6956798*

Filiaal ID: ***56*** **Datum reservering**: *05-09-2001*
Klantnummer: *0078* **Klantnaam**: *Willemsen*
Adres: *steenstraat 5, Zoetermeer* **Postgebied:** *4567 hj*
Telefoonaansluiting: *0031-04567-45678*
Rijbewijs: *hjk-89670-vbngh* **Type Rijbewijs**: *BE*
Model: *Opel Astra* **Kenteken:** *45-gh-23*
Te huren van: *07-09-2001* **tot en met:** *09-09-2001*
Afhaaltijd na 08.00 uur. Terugbrengtijd voor 19.00 uur
Te betalen: *€ 225,-* **Terugbrengstatus:** *op tijd* **Transitfiliaal:** *53*
Tankinhoud: *Vol* **Kilometerstand:** *345678*

Fig. 6: voorbeeld (data use case) huurovereenkomst

[1] Voor sommige huurovereenkomsten is het mogelijk dat er *geen* uiterste verhuurdatum wordt opgegeven.

2.2 Modellering van het informatieperspektief in Kenniskunde

Het informatieperspektief in Kenniskunde wordt gemodelleerd door het analyseren van concrete voorbeelden uit het universum van discussie (UvD). Via een nauwgezet stappenplan (zie [11]) kan de analist uit deze concrete voorbeelden tezamen met een of meer gebruikers van deze voorbeelden komen tot een precieze beschrijving van het informatieperspektief van een bedrijfssysteem. Het resultaat van zo'n analyse wordt *applicatie informatie grammatica* genoemd (AIG). Deze AIG beschrijft de toegestane toestanden en toestandsovergangen en de niet-toegestane toestanden en toestandsovergangen van de *applicatie informatie bank* (AIB).

Begrip	**Bijbehorend naamklasse**	**Definitie van naamsconventie**
Klant	klantnummer	Kan worden gebruikt om een specifieke klant te identificeren binnen de verzameling van alle klanten van Herst
Datum	datumcode	Kan worden gebruikt om een specifieke datum te identificeren binnen de verzameling van alle datums
Rijbewijs	rijbewijscode	Kan worden gebruikt om een specifiek rijbewijs te identificeren binnen de verzameling van alle rijbewijzen
Adres	adrescode	Kan worden gebruikt om een specifieke adres te identificeren binnen de verzameling van alle adressen
Telefoonaansluiting	telefoonnummer	Kan worden gebruikt om een specifieke telefoonaansluiting te identificeren binnen de verzameling van alle telefoonaansluitingen
Postgebied	postcode	Kan worden gebruikt om een specifiek postgebied te identificeren binnen de verzameling van alle postgebieden

Fig. 7: lijst van naamsconventies voor begrippen uit het UvD van figuur 2

Een applicatie informatiegrammatica bestaat uit een begrippenlijst, een lijst van naamsconventies voor de elementen van de genoemde begrippen, voorzover deze begrippen geen werkwoorden zijn, feittypen, en zinsjablonen die aangeven hoe de bijbehorende zinnen dienen te worden uitgesproken en beperkingregels op de populaties van - en de overgangen tussen- populaties van deze feittypen. We hebben de begrippenlijst hier gemakshalve weggelaten, maar de lijst van naamsconventies voor de bergippen uit figuur 1 is weergegeven in figuur 7. Als we de begrippen uit het UvD van figuur 1 nogmaals de revue laten passeren dan zien we dat voor de begrippen *Verhuurverleden* en *Rijbewijstype* geen definitie van een naamsconventie is opgenomen in het overzicht van figuur 7. De reden hiervoor is dat de mogelijke waarden die gebruikt kunnen worden om instanties van deze begrippen aan te duiden expliciet worden vermeld in de waarderegels *c2* en *c4* zoals weergegeven in figuur 8. De AIG van het gebruikersvoorbeeld uit figuur 2 is weergegeven in figuur 8 en bestaat uit 3 feittypen en 7 populatiebeperkingregels. Verder is van elke rol die niet wordt bedekt door een uniciteitsregel aangegeven of het voorkomen van een waarde in deze rol verplicht is (NL=niet leeg) of optioneel (=OP). De rolaanduiding NL in de rollen *klantnaam, datum, rijbewijs, adres* en *verhuurverleden* van feittype *Ft2* geeft aan dat van een klant tenminste één klantnaam, tenminste één indienstredingsdatum, tenminste één rijbewijs, tenminste één klantadres en tenminste één verhuurverleden moet worden vastgelegd. De rolaanduiding OP van de rol telefoonaansluiting uit feittype *Ft2* geeft aan dat het registreren van een klantentelefoonaansluiting *niet* verplicht is. Beperkingsregel *c1* is een uniciteitsregel en geeft aan dat een klant ten hoogste één klantnaam kan hebben, ten hoogste één indiensttredingdatum bij Herst kan hebben, ten hoogste één rijbewijs kan hebben, tenhoogste één adres kan hebben, ten hoogste één telefoonaansluiting kan hebben en tenhoogste één verhuurleden kan hebben. Beperkingsregel *c2* is een waarderegel en geeft aan dat de mogelijke waarden voor het verhuurverleden zijn beperkt tot *goed* of *fout*. Beperkingssregel *c3* is een gelijkheidsregel en geeft aan dat een klant een rijbewijs dient te bezitten dat minimaal één rijbewijstype heeft. Beperkingssregel *c4* is wederom een waarderegel en geeft aan dat het rijbewijstype de waarde BE, CE of DE dient te hebben. Uniciteitsregel *c5* geeft aan dat het rijbewijs

van een klant meerdere rijbewijstypen kan hebben en dat een bepaald rijbewijstype bij meerdere klanten kan voorkomen. De rolaanduiding NL in de rol postgebied van het feittype *Ft1* geeft aan dat voor elk adres een postgebied dient te worden geregistreerd. Deelverzamelingsregel *c6* geeft aan dat als een adres bekend is eveneens de postcode van het postgebied bekend dient te zijn. Uniciteitsregel *c7* geef tenslotte aan dat een adres deel uit kan maken van ten hoogste één postgebied.

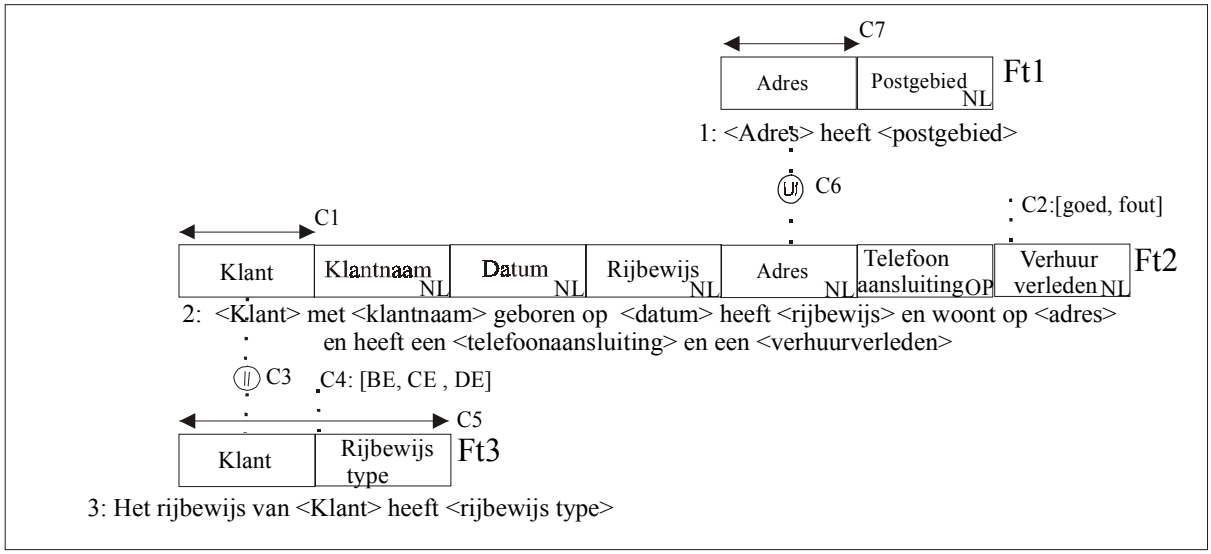

Fig. 8: applicatie informatiegrammatica (AIG) van klantoverzicht uit figuur 2

In figuren 9 en 10 staan de ontbrekende definities van begrippen en naamsconventies voor de UvD die bestaat uit de voorbeelden in figuren 2 tot en met 6.

Begrip	**Definitie van het Begrip**
Auto	Een voertuig geregistreerd door de Nederlandse overheid.
Leenauto	Een auto die Herst kan gebruiken voor verhuur
Model	Een karakterisering van een auto.
Bedrag	Een hoeveelheid geld
Onderhoudstatus	Een onderhoudstatus van een auto geeft aan hoeveel kilometers de auto op dat moment heeft gereden vanaf de laatste onderhoudsbeurt.
Filiaal	Een vestiging van Herst autoverhuur waar klanten autos kunnen huren.
Huurovereenkomst	Een wettelijke overeenkomst tussen Herst en een Klant waarin de voorwaarden en afspraken m.b.t. het huren van een auto zijn opgenomen.
Werknemer	Een persoon die werkzaam is bij Herst
Opleiding	Een indicator van de kennis die een werknemer bezit.
Specialisatie	Een indicator van de ervaring die een werknemer heeft.
Commissie	Een percentage van de omzet die als additionele betaling aan een werknemer jaarlijks wordt uitgekeerd
Kilometerstand	Een indicator van het aantal kilometers dat een auto heeft afgelegd.
Tankinhoud	Een indicator die aangeeft of een auto een volle of een niet-volle benzinetank heeft

Fig. 9: Additionele definities van begrippen uit het UvD van figuren 3 tot en met 6

Begrip	Bijbehorend naamklasse	Definitie van naamsconventie
Leenauto	leenautonummer	Kan worden gebruikt om een specifieke leenauto te identificeren binnen de verzameling van alle leenautos van Herst
Auto	kenteken	Kan worden gebruikt om een specifieke auto te identificeren binnen de verzameling van alle auto's in Nederland.
Model	modelnaam	Kan worden gebruikt om een specifiek model auto te identificeren binnen de verzameling van alle automodellen
Bedrag	geheel getal	Kan worden gebruikt om een specifiek bedrag in euro's te identificeren binnen de verzameling van alle bedragen in euro's
Onderhoudstatus	geheel getal	Kan worden gebruikt om een specifieke onderhoudstatus te identificeren binnen de verzameling van alle onderhoudstatussen
Filiaal	filiaal ID	Kan worden gebruikt om een specifiek filiaal te identificeren binnen de verzameling van alle filialen van Herst in Nederland.
Huurovereenkomst	huurovereenkomstcode	Kan worden gebruikt om een specifieke huurovereenkomst te identificeren binnen de verzameling van alle huurovereenkomsten van Herst in Nederland.
Werknemer	werknemercode	Kan worden gebruikt om een specifieke werknemer te identificeren binnen de verzameling van werknemers die nu werken of ooit gewerkt hebben voor Herst in Nederland
Opleiding	opleidingscode	Kan worden gebruikt om een specifieke opleiding te identificeren binnen de verzameling van alle opleidingen.
Specialisatie	specialisatiecode	Kan worden gebruikt om een specifieke specialisatie van een monteur te identificeren binnen de verzameling van alle specialisaties voor een monteur.
Commissie	commissiepercentage	Kan worden gebruikt om een specifieke commissie te identificeren binnen de verzameling van commissies
Kilometerstand	geheel getal	Kan worden gebruikt om een specifieke kilometerstand te identificeren binnen de verzameling van alle kilometerstanden

Fig. 10: Additionele naamsconventies voor de begrippen uit het UvD van figuren 3 tot en met 6

Voor de begrippen *verhuurstatus* en *terugbrengstatus* uit de integrale UvD van figuren 2 tot en met 6 wordt geen naamsconventie opgenomen omdat alle mogelijke waarden die naar zo'n status verwijzen reeds zijn opgenomen bij de toestandsovergangs-beperkingsregel *c10,* respectievelijk waarderegel *c25* in figuur 11. In figuur 11 hebben we de totale informatiegrammatica voor het universum van discussie (dat bestaat uit figuren 2 tot en met 6) gegeven.

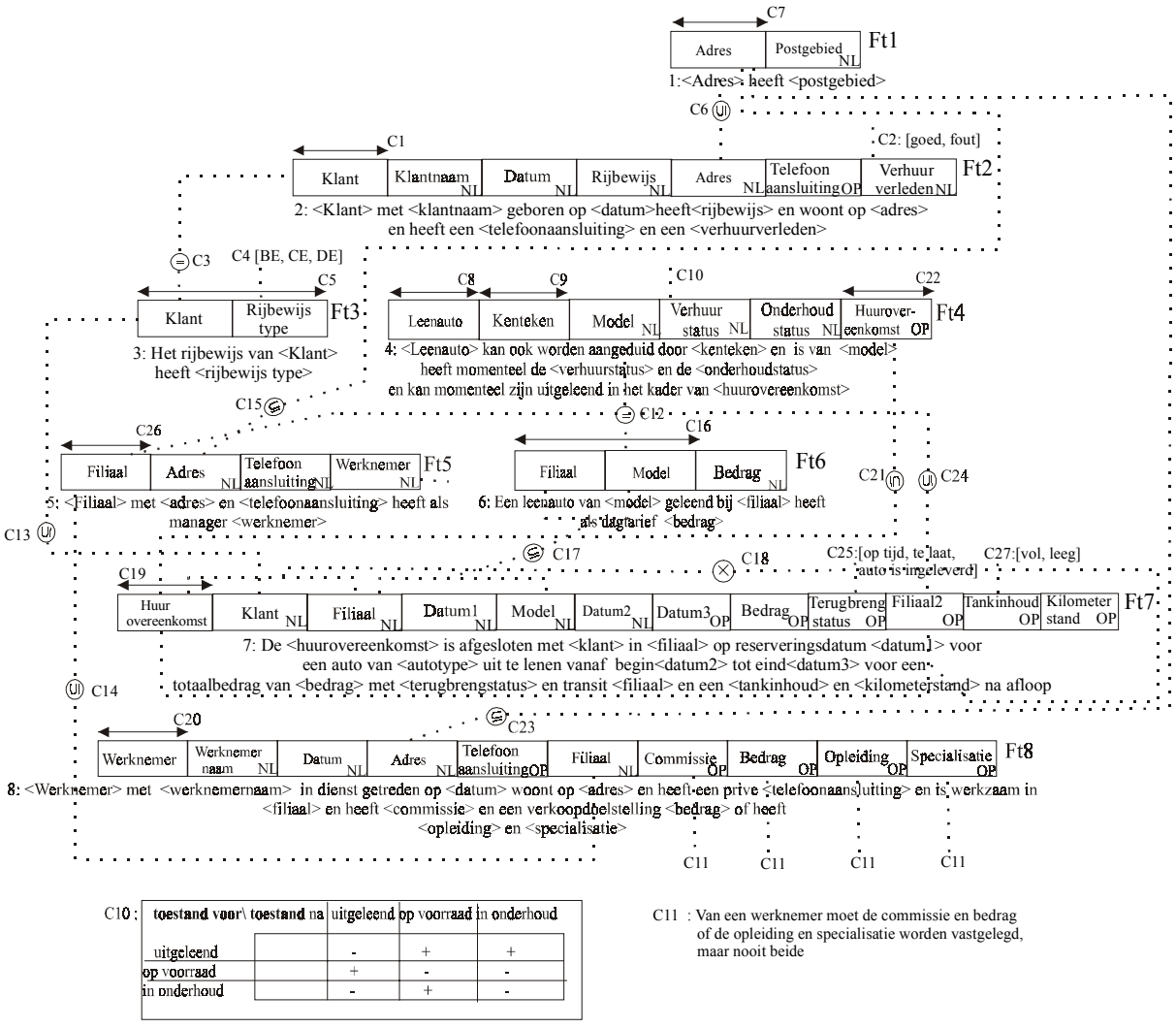

Fig. 11: complete applicatie informatiegrammatica (AIG) van verhuur casus

Wat ons verder opvalt in de AIG van figuur 11 is dat er een specialisatie bestaat tussen de concepten *leenauto* en *auto.* Een leenauto is een auto, en derhalve is het mogelijk om de naamsconventie van het meest algemene concept, in dit geval *auto* te gebruiken als (alternatieve) naamklasse voor het concept *leenauto*. Met de rollen *leenauto* en *kenteken* van feittype *Ft4* in de applicatie informatiegrammatica van figuur 11 wordt dit geïllustreerd. De reden dat Herst een bedrijfsspecifiek leenautonummer heeft ingevoerd is dat deze nummers makkelijker te communiceren zijn. We zullen nu een deel van de beperkingsregels uit het informatiegrammaticadiagram van figuur 11 bespreken.

Uniciteitsregels *c8* en *c9* in figuur 11 geven aan dat een specifieke leenauto *ten hoogste* één model kan hebben, *ten hoogste* één verhuurstatus heeft en *ten hoogste* één onderhoudstatus heeft. De roleigenschappen NL voor de rollen *model, verhuurstatus* en *onderhoudstatus* van feittype *Ft4* geven aan dat bij elke leenauto *minimaal* één model behoort, er *minimaal* één verhuurstatus is en er *minimaal* één onderhoudstatus is. Beperkingsregel c*10* in figuur 11 is een voorbeeld van een toestandsovergangsbeperkingsregel. In de bijbehorende matrix staan de mogelijke waarden, respectievelijk toestanden waarin een leenauto zich kan bevinden. Een – betekent dat een mogelijk toestandsovergang niet is toegestaan en een + betekent dat zo'n toestandsovergang wel is toegestaan. Zo kan bijvoorbeeld een auto nooit vanuit de toestand in onderhoud in de toestand uitgeleend komen. Deelverzamelingsregel *c13* geeft aan dat elke klant die een huurovereenkomst heeft, minimaal één rijbewijs dient te bezitten. Gelijkheidsregel *c12* geeft aan dat voor elk model leenauto dat door Herst wordt aangeboden er tenminste één leenauto in de vloot moet zijn opgenomen. De samengestelde deelverzamelingsregel *c17* geeft aan dat een leenauto die is uitgeleend voor een huurovereenkomst

altijd in de vloot dient te zijn opgenomen. De algemene beperkingsregel *c11* gedefinieerd op de rollen *commissie, bedrag, opleiding* en *specialisatie* van het feittype *Ft8* geeft aan dat van een werknemer oftewel de commissie en het bedrag van de verkoopdoelstelling worden vastgelegd of dat er één *opleiding* en één *specialisatie* worden opgeslagen. De waarderegel *c25* geeft aan dat een terugbrengstatus uitsluitend één van de de waarden *op tijd, te laat* of *auto is ingeleverd* kan hebben. De deelverzamelingsregel *c24* geeft aan dat een transitfiliaal een bestaand Herst filiaal moet zijn. Uitsluitingsregel *c18* die is gedefinieerd over twee rollen aan dat bij dezelfde verhuurovereenkomst het transitfiliaal altijd moet afwijken van het filiaal waar de verhuurovereenkomst is afgesloten.

In sectie 2.2 hebben we een voorbeeld gegeven hoe het informatieperspektief in Kenniskunde wordt gemodelleerd. In sectie 2.3 zullen we het procesperspektief dat behoort tot ons voorbeeld modelleren.

2.3 Modellering van het procesperspektief in Kenniskunde

Het informatieperspektief bechrijft 'wat' er gecommuniceerd wordt in het UvD. ' Hoe' deze gecommuniceerde feiten tot stand komen vormt het aandachtsgebied van het procesperspektief. Het procesperspektief wordt in Kenniskunde afgebeeld middels procestypen. Een *procestype* bevat onder meer afleidingsregel(s) die beschrijven hoe rolinstanties kunnen worden afgeleid uit andere rolinstanties van de AIG die behoort tot de UvD. Als we de AIG die hoort bij het autoverhuur voorbeeld als uitgangspunt nemen kunnen we concluderen dat een waarde in de rol *bedrag* van feittype *Ft7* altijd kan worden afgeleid door de waarden in de rol *geldbedrag* van een feitinstantie van het feittype *Ft6* waarvan de combinatie van filiaal en model overeenkomt met de filiaal van de huurovereenkomst (de invulplaats of rol *filiaal* in feittype *Ft7*) en waarvan de leenauto het model heeft uit de invulplaats (of rol) *model* van feittype *Ft7* te vermenigvuldigen met het verschil tussen de waarde van de invulplaats *datum3* uit feittype *Ft7* en de invulplaats *datum2* uit feittype *Ft7*[2]. De Kenniskunde *procesbeschrijving* voor dit procestype is gegeven in figuur 12.

Procestype_1_ huurovereenkomstbedrag(arg1:huurovereenkomst, arg2:filiaal, arg3:model)
***Preconditie**: 'arg1'∈ FT7.huurovereenkomst*
***Afleidingsregel AR1**:*
*FT6.geldbedrag [waar FT6.model='arg3' EN FT6.filiaal='arg2')] **
(FT7.datum3[waarFt7.huurovereenkomst=arg1]–FT7.datum2
[waar Ft7.huurovereenkomst=arg1]
***Potsconditie** creëer instantie van FT7.bedrag*
[waar FT7.huurovereenkomst='arg1' EN FT7.bedrag= AR1]

Fig. 12: Procesbeschrijving voor het berekenen van totaalbedrag uit huurovereenkomst

In het voorbeeld van de procesbeschrijving in figuur 12 zien we dat het proces allereerst checkt in de *pre-conditie* van het procestype of de huurovereenkomst waarvan het bedrag dient te worden berekend überhaupt bestaat. Verder dienen er argumenten gespecificeerd te worden die aangeven voor welk model auto en in welke filiaal een huurovereenkomst dient te worden opgesteld, omdat anders geen dagtarief kan worden gevonden in de applicatie informatiebank. Als de huurovereenkomst niet bestaat kan het proces dus niet worden uitgevoerd. Indien er wordt voldaan aan de pre-conditie van het proces kan de afleidingsregel worden toegepast. In de *post-conditie* van het procestype wordt tenslotte aangegeven hoe de waarde(n) die resulteert(en) uit het proces er uitziet(n). In figuur 13 wordt nog een voorbeeld van een proces uit de applicatie procesbeschrijving van het autoverhuurvoorbeeld gegeven.

[2] We gaan er hier vanuit dat de begin- en einddatum nooit dezelfde zijn.

Procestype_2_voegtoe_terugbrengstatus(arg1:huurovereenkomst,arg2:datum)
***Preconditie**: 'arg1'∈ FT7.huurovereenkomst*
***Afleidingsregel AR2**:*
ALS FT7.datum3[WAAR FT7.huurovereenkomst=arg1] BESTAAT
DAN ALS 'arg2'>= FT7.datum3[WAAR FT7.huurovereenkomst=arg1]
DAN AR2:=te laat
ANDERS AR2:= op tijd
EINDALS
ANDERS AR2:=auto is ingeleverd
EINDALS
***Postconditie** creëer instantie van FT7.terugbrengstatus [waar FT7.huurovereenkomst='arg1'*
EN FT7.terugbrengstatus:= AR2]
***EN** creëer instantie van FT7.tankinhoud*
***EN** creëer instantie van FT7.kilometerstand*

Fig. 13: Procesbeschrijving voor toevoegen terugbrengstatus

Procestype 2 in figuur 13 is verantwoordelijk voor het creëren van waarden voor de invulplaatsen *terugbrengstatus, tankinhoud en kilometerstand* van het feittype *Ft7*. Als we het deel van de applicatie informatiegrammatica van de verhuurovereenkomst in figuur 9 inspecteren zien we dat er slechts 3 waarden zijn toegestaan in de rol Ft7.terugbrengstatus: *op tijd, te laat* en *auto is ingeleverd*. Het spreekt voor zich dat de mogelijke uitkomsten van een proces van procestype 2 aan zo'n populatiebeperkingsregel dienen te voldoen. Als we de afleidingsregel AR2 goed bekijken dan zien we dat voor alle mogelijke combinaties van de procesargumenten, het proces een verhuurstatus zal creëren die overeenkomt met een van de waarden in de beperkingsregel c31. Procestype 2 kan worden beschouwd als de formalisering van de volgende bedrijfsregel uit het autoverhuurbedrijf:

*Als een auto wordt ingeleverd wordt gecheckt of de bijbehorende huurovereenkomst bestaat. Zo, ja dan wordt de huidige datum vergeleken met de datum op de verhuurovereenkomst. Als de beide data overeenkomen of indien de werkelijke datum vroeger is dan de datum op de verhuurovereenkomst dan wordt de terugbrengstatus gelijk gesteld aan **op tijd**. Als de huidige datum later is dan de datum in de verhuurovereenkomst dan wordt de terugbrengstatus: **te laat**. Als er geen einddatum op de verhuurovereenkomst staat wordt de verhuurstatus **: auto is ingeleverd**. Tevens wordt de tankinhoud van de teruggebrachte auto gecontroleerd. Als de tank helemaal vol is wordt de tankinhoud op **vol** gesteld in alle andere gevallen wordt de waarde **leeg** toegekend. Tenslotte wordt de kilometerstand opgenomen en geregistreerd op de verhuurovereenkomst.*

Nadat de processen uit de procesbeschrijving zijn uitgevoerd worden de resultaten voorgelegd aan de *conceptuele informatie processor* [12] en wordt er gekeken of in de zo verkregen toestand van de applicatie informatiebank een of meer populatie- en/of populatie-overgangsbeperkingsregels worden geschonden. Als dit het geval is dan worden de betreffende feitinstanties niet toegevoegd aan, c.q. verwijderd uit de applicatie informatiebank. Indien geen populatie- of populatieovergangs-beperkingsregels worden geschonden kunnen de resultaten worden verwerkt in de applicatie-informatiebank. Dit betekent dat de AIG alle toegestane en niet-toegestane toestanden van de AIB voorschrijft. Ook de mogelijke resultaten van het uitvoeren van processen uit de procesbeschrijving dienen de AIG te respecteren.

2.4 Modellering van het gedragsperspektief in Kenniskunde

De laatste modelsoort die wordt onderkend in Kenniskunde op het niveau van applicaties is de modelsoort die gebruikt wordt voor het beschrijven van het gedragsperspektief van een bedrijfssysteem. In sectie 2.2 hebben we laten zien hoe een applicatie informatiegrammatica beschrijft *wat* er gecommuniceerd kan worden in het bedrijfssysteem. In sectie 2.3 hebben we gezien dat een procesbeschrijving beschrijft *hoe* instanties van de feittypen uit de applicatie informatiegrammatica in het bedrijfssysteem worden gecreëerd. Tenslotte zullen we in deze sectie de modelsoort beschrijven die weergeeft *wanneer* een proces uit de procesbeschrijving in werking kan worden gezet in het

bedrijfssysteem. Deze modelsoort noemen we de *eventbeschrijving*. Een eventbeschrijving bestaat uit eventregels die weergeven onder welke conditie op de applicatie informatiebank het optreden van een event leidt tot het opstarten van een bepaald proces uit de procesbeschrijving.

2.4.1 Interne events

Er bestaan events die samen vallen met gebeurtenissen binnen het bedrijfssysteem, bijvoorbeeld wanneer een feit wordt toegevoegd aan de informatiebank. Deze eventsoort noemen we *interne events*. Niet alle interne events zullen leiden tot het opstarten van een proces uit de procesbeschrijving. We zijn derhalve uitsluitend geïnteresseerd in die events die leiden tot het opstarten al dan niet onder voorwaarden van een proces uit de procesbeschrijving. Een voorbeeld van zo'n event is het toevoegen van een waarde van de rol *datum3* van feittype *Ft7* in de informatiebank. Als we dit interpreteren in termen van het applicatiedomein kunnen we zeggen dat het moment waarop dit event plaatsvindt kan worden geïnterpreteerd als het bekend worden van de verhuurtermijn van de auto in de huurovereenkomst. We weten dat dit event het berekenen van het totaalbedrag van de huurovereenkomst initieert.

WANNEER instantie_van_rol_datum3 van feittype ft7_wordt_toegevoegd(arg:huurovereenkomst)
DAN Proces_1_ huurovereenkomstbedrag(arg1:huurovereenkomst, arg2:filiaal,arg3:autotype)
Impuls-parameter-toewijzer:
Proces_1_huurovereenkomstbedrag.arg1:=
instantie_van_rol_datum3 van feittype ft7 _wordt_toegevoegd.arg
Proces_1_huurovereenkomstbedrag.arg2:=Ft7.filiaal
[WAAR Ft7.huurovereenkomst=
instantie_van_rol_datum3 van feittype ft7 _wordt_toegevoegd.arg]
Proces_1_huurovereenkomstbedrag.arg3:=Ft7.autotype
[WAAR Ft7.huurovereenkomst=
instantie_van_rol_datum3 van feittype ft7 _wordt_toegevoegd.arg]

Fig.14: Voorbeeld van een event regel

In figuur 14 hebben we deze event-regel formeel weergegeven. In deze eventregel wordt er onconditioneel een huurovereenkomstbedrag berekend. De impulsparameter-toewijzer geeft precies aan voor welke huurovereenkomst, het totaalbedrag dient te worden berekend. De argumentwaarden voor het proces worden hier geïnstantieerd. De eerste argumentwaarde is gelijk aan de waarde van de huurovereenkomst in de rol *datum3* van feittype *Ft7* dat wordt toegevoegd aan de informatiebank. De tweede argumentwaarde voor het proces wordt verkregen door in de informatiebank op te zoeken in welke filiaal de huurovereenkomst is afgesloten. De derde argumentwaarde tenslotte wordt verkregen door in de informatiebank het autotype op te zoeken dat hoort bij deze huurovereenkomst.

2.4.2. Externe events

Een ander soort events zijn de events die van buiten het bedrijfssysteem komen, deze worden *externe events* genoemd. Een voorbeeld van een extern event is het terugbrengen van een huurauto door een klant. In dit voorbeeld gaan we er vanuit dat het er niet toe doet in termen van de 'werkelijkheid' hoe het terugbrengen plaatsvindt. Het enige dat voor de gedragsbeschrijving relevant is dat een huurauto die hoort bij een huurovereenkomst wordt teruggebracht. De argumentwaarden die worden meegeven aan dit event zijn de waarden voor *huurovereenkomst*, het *kenteken* en de werkelijke *datum* van terugbrengen. We gaan er vanuit dat de verhuuragent er op toeziet dat de teruggebrachte auto inderdaad die auto is die dient te worden teruggebracht. Indien niet het geval is kan de huurovereenkomst nooit een terugbrengstatus krijgen. In figuur 15 hebben we deze event-regel weergegeven. We zullen dit event type *terugbrengen* noemen.

WANNEER terugbrengen(arg1:huurovereenkomst, arg2:leenautokenteken, arg3:datum)
ALS terugbrengen.arg1 ∈ EXT(Ft7.huurovereenkomst)
EN terugbrengen.arg2= FT4.kenteken [waar FT4.huurovereenkomst='arg1']
DAN Proces_2_voegtoe_terugbrengstatus(arg1:huurovereenkomst,arg2:datum)
Impulsparameter-toewijzer:
Proces_2_voegtoe_terugbrengstatus.arg1:= terugbrengen.arg1
Proces_2_voegtoe_terugbrengstatus.arg2:= terugbrengen.arg3

Fig.15: tweede voorbeeld van een event regel

We zien in figuur 15 hoe de conditie *ALS terugbrengen.arg1 ∈ EXT (Ft7.huur-overeenkomst) EN terugbrengen.arg2= FT4.kenteken [waar FT4.huurovereenkomst='arg1']* gebruik maakt van de toestand van de applicatie-informatiebank. De eerste regel van de impulsparameter-toewijzer illustreert hoe wordt aangegeven *welk* proces van type 2 dient te worden uitgevoerd namelijk het toevoegen van de terugbrengstatus voor een *specifieke* huurovereenkomst. De tweede regel illustreert wat de precieze terugbrengdatum is die dient te worden meegegeven bij de aanroep van het proces.

We kunnen nu concluderen dat de 3 perspektieven in Kenniskunde gebruik maken van elkaars resultaten. Die afhankelijkheid impliceert echter een specifieke volgorde die men dient te respecteren bij het conceptueel modelleren. Eerst dient men de applicatie-informatiegrammatica (AIG) op te stellen, vervolgens de procesbeschrijving (PB) en tenslotte de eventbeschrijving (EB). We hebben in deze paragraaf laten zien hoe de modellen van Kenniskunde alle informatiekundige aspecten (nl. wat, hoe en wanneer) van een bedrijfssysteem beschrijven. In sectie 3 zullen we de verschillende modelsoorten in de Unified Modeling Language (UML) [1, 14, 17] in hun onderlinge samenhang beschrijven. In sectie 4 geven we aan hoe de verschillende onderdelen uit de Kenniskunde modelsoorten afgebeeld kunnen worden in de UML diagramsoorten.

3. Diagramsoorten in UML

De modelsoorten in Kenniskunde zijn gericht op domeinaspecten van het bedrijfssysteem en kunnen derhalve beschouwd worden als conceptuele modellen. Als we spreken over de implementatie van de domeinkennis in webpaginas of geautomatiseerde informatiesystemen in het algemeen introduceren we specifieke oplossingsdomeinkennis: namelijk de kennis van de hardware van het geautomatiseerde informatiesysteem of kennis van de softwaremodules van het informatiesysteem. In principe kunnen we voor het modelleren van deze domeinen dezelfde modelleringconstructies gebruiken. Het enige wat er in zo'n geval verandert is het universum van discussie. Aangezien in Kenniskunde het altijd duidelijk is *wat* er gemodelleerd dient te worden kan men ook zeggen dat men voor het beschrijven van de implementatie-aspecten, een applicatie informatiegrammatica, een procesbeschrijving en een eventbeschrijving kan maken waarbij de uitgangsvoorbeelden worden gevormd door voorbeelden van softwarearchitectuur, netwerkconfiguraties enzovoort. Het enige wat er verandert in zo'n geval is het universum van discussie. In UML wordt deze logica *niet* toegepast en definieert men specifieke modelleringsconstructies voor sommige implementatie-aspekten. In deze paragraaf zullen we de 9 modelsoorten die UML kent beschrijven en zullen we tijdens deze bespreking de samenhang van de verschillende modelsoorten expliciet aangeven. In figuur 16 staan de 9 modelsoorten van UML weergegeven.

Klassendiagram
Objekt diagram
Use-case diagram
Sequence diagram
Collaboration diagram
Toestandsdiagram
Activiteitendiagram
Component diagram
Deployment diagram

Fig. 16: Modelsoorten in UML

Een tekortkoming in de bestaande literatuur over UML is dat men er niet in is geslaagd om de samenhang tussen de genoemde diagramsoorten expliciet te maken.

In het vervolg van dit artikel zullen wij de diagramsoorten in UML die nodig zijn om het *informatieperspektief*, het *procesperspektief* en het *gedragsperspektief* van een bedrijfssysteem volledig te beschrijven op conceptueel niveau in hun onderlinge samenhang behandelen. In het vervolg van deze sectie zullen we laten zien hoe de modelelementen uit de *applicatie informatie grammatica*, de *procesbeschrijving* en de *eventbeschrijving* binnen Kenniskunde afgebeeld worden op de respectievelijke modelelementen van de UML diagramsoorten en zullen we de samenhang tussen de UML diagramsoorten laten zien. We zullen ook de tekortkomingen van deze diagramsoorten in het kader van de conceptuele bedrijfsmodellering aantonen.

3.1 Het use-case diagram

Het doel van een *use case* zoals momenteel wordt beschreven in de definiërende UML literatuur [1, 14, 17] is om de namen van de diensten die door een applicatiesysteem worden verricht te beschrijven in termen zoals die door de gebruikers worden ervaren. We merken op dat use cases derhalve *niet* gebruikt dienen te worden om de inhoud van deze diensten te beschrijven. De definiërende UML literatuur [17] laat helaas niet de totale samenhang zien tussen de *use-cases* en de *overige* UML diagramsoorten.

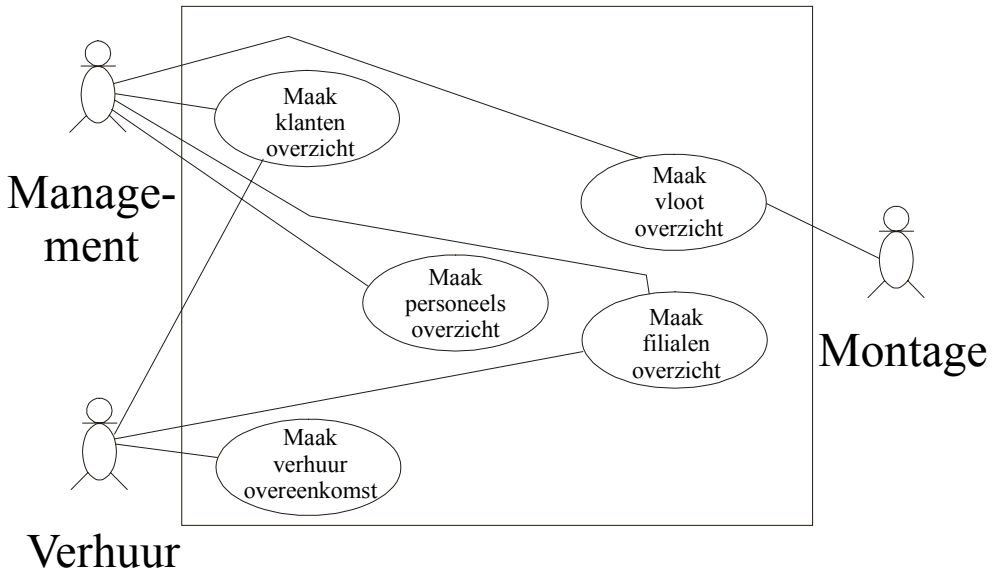

Fig.17: use case voorbeeld

Een use case diagram laat zien wat de namen van de diensten zijn die door het applicatie-informatiesysteem kunnen worden aangeboden[3] aan welk type gebruiker in een bepaalde rol (oftewel actor). Welke *actoren* men kan onderkennen in een specifiek applicatiegebied wordt echter niet aangegeven in UML. Ook wordt niet aangegeven hoe men komt tot de afbakening van *use-cases*. We zullen nu vanuit het UvD en de casusbeschrijving van Herst Autoverhuur proberen te komen tot het afleiden van de use cases en de actoren die gebruik maken van deze use cases. Het is belangrijk om in te zien dat het concept van *actor* afwijkt van het concept *gebruiker*. We zullen dit laten zien aan de hand van het autoverhuur voorbeeld. Herst Nederland heeft een zestigtal filialen. Deze filialen variëren in omvang van 1-persoons filialen tot vestigingen waar 8 mensen werkzaam zijn. Zo bestaan er vestigingen waar de filiaal manager *geen* specifieke verhuurtaken verricht. In de 1-persoonsfilialen daarentegen is de enige werknemer tevens filiaalmanager, die voor het grootste deel van zijn tijd verhuuragent werkzaamheden verricht. Het verdient derhalve aanbeveling om bij het modelleren van actoren, namen te gebruiken die **niet** overeenkomen met de namen van functies. In het voorbeeld van

[3] We merken op dat de inhoud van deze diensten niet kan worden beschreven in een Use case diagram.

figuur 17 kan men nu makkelijk inzien dat bij een één-persoons filiaal de filiaalmanager (annex verhuuragent) de actoren 'management' en 'verhuur' speelt. In het geval van een 'groot' filiaal speelt de filiaalmanager de actor 'management' en wordt de actor 'verhuur' gespeeld door één of meer verhuuragenten. In figuur 17 hebben we een voorbeeld gegeven waarbij we slechts drie actoren in het systeem onderkennen: *management*, *verhuur* en *montage*. In het voorbeeld van figuur 15 kan men zien dat actoren van het actortype *management* de diensten *maak klanten overzicht, maak vlootoverzicht maak filialenoverzicht, maak personeelsoverzicht* van het applicatiesysteem kunnen opvragen. Een actor van het actortype *verhuur* kan de diensten *maak klantenoverzicht*, *maak filiaaloverzicht* en *maak verhuurovereenkomst* opvragen. Tenslotte kunnen actoren van het type *montage* de dienst *maak vlootoverzicht* opvragen. Een voorbeeld van een instantie van de use case uit figuur 17 is : *Verhuuragent **1017** roept de use case maak verhuurovereenkomst **6956798 op 05-09-2001 te 17.15** aan.* Het is meteen duidelijk dat het *resultaat* van elk van deze 5 diensten of use cases samenvalt met een instantie van een document uit figuren 2 tot en met 6. Dit is de reden dat deze documentensoorten dan ook *data use-cases* worden genoemd [8, p.104]. Het is gemakkelijk in te zien dat *use cases* overeenkomen met het resultaat van het uitvoeren van (samengestelde) processen (bijvoorbeeld het reserveren van een auto) en het zich voordoen van events (bijvoorbeeld het event dat een klant een auto terug brengt) die leiden tot aanpassingen op de data-use cases.

Als uitbreiding van het use-case concept kent UML *sub-use cases*. Het zal duidelijk zijn dat in het geval van ons voorbeeld, de use-case *maak verhuurovereenkomst*, de resulterende data-use case op verschillende momenten een verschilllende inhoud heeft. Men kan op grond van deze verschillende data-inhouden 3 sub-use cases onderscheiden: *maak reservering*, *haal leenauto op* en *breng leenauto terug* (zie figuur 18).

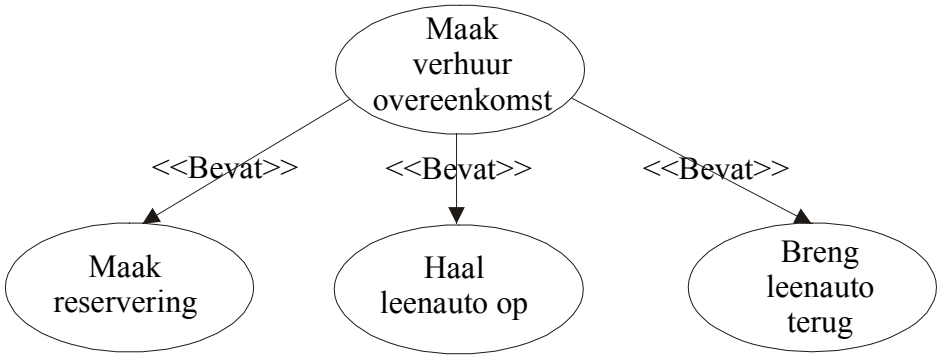

Fig.18: sub-use case voorbeeld

Als we nu de AIG die de inhoud van een gegeven data-use case voorschrijft goed analyseren, dan kunnen we gemakkelijk vaststellen of er (een) *sub-use case(s)* te vinden zijn (is), die deel uitma(a)k(t)en van de generieke use-case (bijvoorbeeld *maak verhuurovereenkomst*). Zo zien we aan het optioneel zijn van sommige rollen in feittype *Ft7* en de deelverzamelingsregel *c17* in figuur 11, dat er tenminste 2 sub-use cases bestaan. Als resultaat van de eerste sub-use case (*maak reservering*) dienen de waarden in de rollen *klant, filiaal, datum1, model* en *datum* van feittype *Ft7* gespecifieerd te worden. De overige rollen die verwijzen naar deze data-use case kan men op enig later moment invullen. De additionele informatie die in UML in een use case diagram wordt weergegeven is welke actor gebruik maakt van welke use case.

3.2. Het klassendiagram en het objektdiagram in UML

Het UML klassendiagram bestaat uit objektklassen en associaties tussen deze objektklassen. Een objektklasse bestaat uit twee delen. Het deel boven de streep bevat de klassenattributen en eventueel de attribuutmultipliciteit. Verder kan van elk klasse-attribuut worden aangegeven tot welk datatype of tot welke objektklasse de (atomaire) waarden van het attribuut behoren. Onder de streep van een objektklasse kan men de klassenoperaties vermelden. Een *klassenoperatie* is de UML modelleringconstructie die verwijst naar het gedrag die een klasse kan vertonen. De implementatie van

een operatie in UML gebeurt via het concept van *klassenmethode*. Het onderscheid tussen *operatie* en *methode* heeft zijn oorsprong in het concept van overerving, dat het mogelijk maakt om in een klasse hiërarchie de implementatie van een operatie te laten verschillen (polymorfisme).

Het spreekt voor zich dat een klassendiagram in UML, allereerst een model dient te zijn in de zin van regelgeving en grammatica van de initiële *voorbeelden* (zoals geïllustreerd in figuren 2 tot en met 6). Ten tweede dient het klassendiagram een model te zijn voor de *data use-cases* die worden verkregen uit de initiële voorbeelden en de use-cases van het applicatie gebied (geïllustreerd in figuur 17).

De totale samenhang tussen de UML use case diagrammen en de klasse-diagrammen voorwat betreft de modellering van het informatie-perspektief kan nu worden gegeven. De domein concepten en hun onderlinge relaties en beperkingen op hun voorkomens worden gemodelleerd als objektklassen, attributen, associaties, multipliciteiten en tekstuele beperkingsregels in een UML klassendiagram. De precieze samenhang tussen het use-case diagram en het klasse diagram van een bepaalde UvD bevindt zich in de declaratie van klasse-operaties behorende tot de objekt-klassen. Elke gedeclareerde operatie dient tenminste eenmaal voor te komen als een *(sub)use-case* van een use-case diagram van het applicatiegebied of dient aan te worden geroepen in een operatie die is gedeclareerd voor zo'n (sub)-use case. Verder bepaalt de AIG mede of een use-case al dan niet in sub-use cases moet worden onderverdeeld.

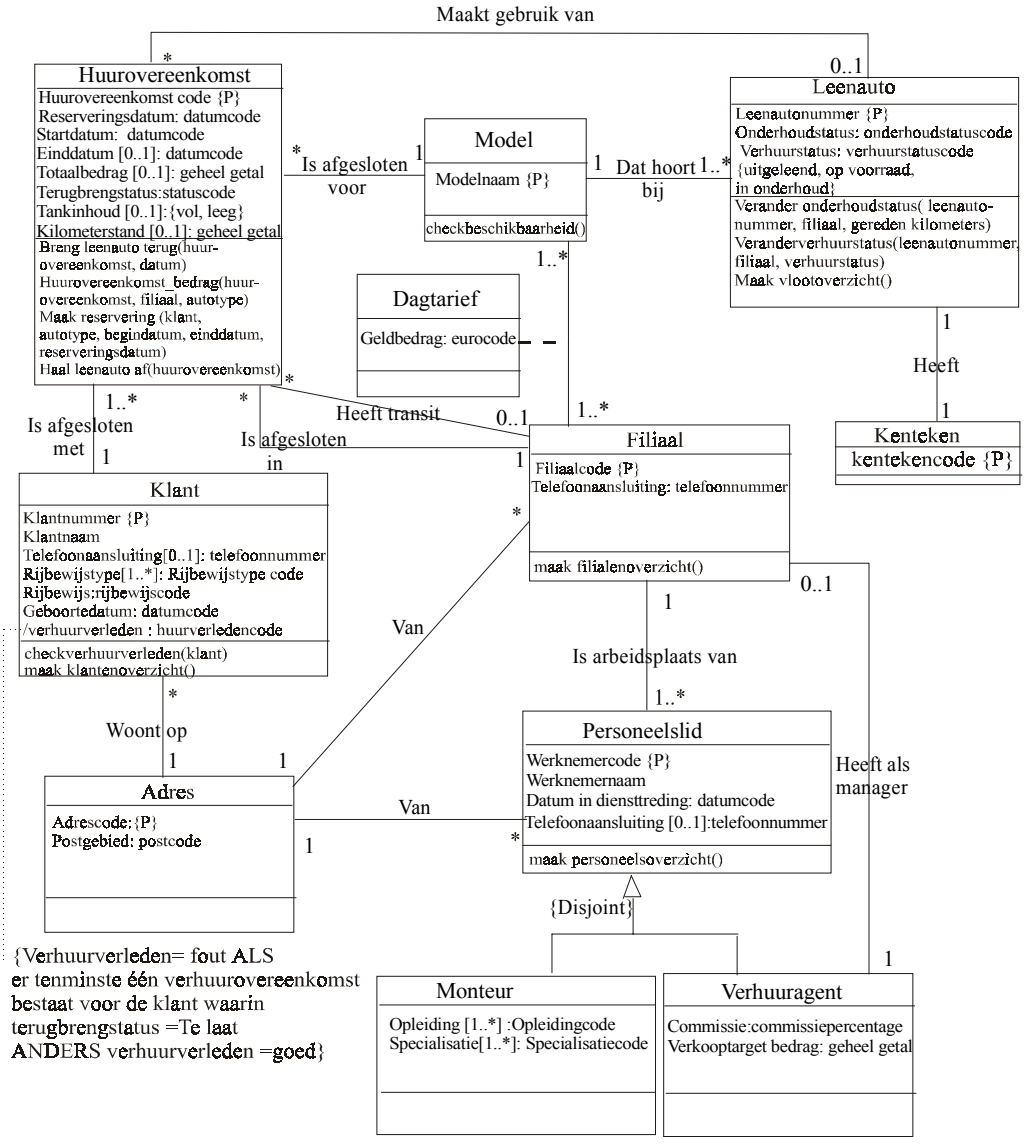

Fig. 19: UML klassendiagram voorbeeld voor integrale autoverhuur UvD

Allereerst laten we de samenhang die het klasse-diagram van het verhuurvoorbeeld in figuur 19 heeft met het bijbehorende (sub)use-case diagram uit figuren 17 en 18 zien d.m.v. de klasse-operatie *maak klantenoverzicht* die gedefinieerd is bij de klasse *Klant*, de klasse-operatie *maak personeelsoverzicht* die gedefinieerd is bij de klasse *Personeelslid*, de klasse-operaties *maak reservering, haal leenauto af, breng leenauto terug, huurovereenkomst-bedrag()* die gedefinieerd zijn bij de klasse *Huurovereenkomst*, de klasse-operatie *maak vlootoverzicht* die gedefinieerd is bij de klasse *Leenauto* en tenslotte de klasse-operatie *maak filialenoverzicht* die is gedefinieerd bij de klasse *Filiaal*.

Vervolgens kunnen we de informatiestructuur van de data-use cases uit het applicatiegebied gaan modelleren in het klassendiagram. Er zijn in hoofdzaak vier manieren in UML om Kenniskunde feittypen te modelleren. Ten eerste kan men *unaire* en atomaire *binaire* feittypen modelleren als attributen [6]. Ten tweede kan men atomaire *binaire* en moet men atomaire *N-aire* feittypen modelleren als associaties [3]. Ten derde kan men unaire feittypen modelleren als UML subklassen. Tenslotte is het mogelijk om N-aire feittypen waarvan N-1 rollen dienen als samengestelde identificatie van een element te modelleren als *associatieklasse* (inclusief klasse-attribuut) in UML. In het klasse-diagram van figuur 19 zien we dat de naamsconventie voor een adres is geëncodeerd in UML als het identificatie attribuut *adres code {P}* van de objektklasse *Adres*. De rol *Adres* van Feittype *Ft2* is geëncodeerd in UML als de associatie *woont op*. De rol klantnaam van Feittype *Ft2* is gemodelleerd als het attribuut *klantnaam* van de objektklasse *Klant* in UML. De roleigenschap *OP* van de rol telefoonaansluiting van feittype *Ft2* geeft aan dat klanten niet altijd beschikken over een telefoonaansluiting. Dit kan worden weergegeven in UML door het klasse attribuut *telefoonaansluiting* van de objektklasse *Klant* optioneel te maken (een minimale attribuut multipliciteit van 0).

De lijst van naamsconventies uit figuren 7 en 10 geeft aan welke naamklasse gebruikt kan worden om instanties van een concept aan te duiden. In UML wordt dit aangegeven door de identificatie attributen *klantnummer (van de klasse Klant)* en *adrescode (van de klasse Adres),* en de attribuuttypen *datumcode* (behorende bij het attribuut *geboortedatum* van de klasse *Klant), postcode (*behorende bij het attribuut *postgebied* van de klasse *Adres*), *telefoonnummer* (behorende bij het attribuut *telefoonaansluiting* van de klasse *Klant, rijbewijs code* (behorende bij het attribuut *rijbewijs* van de klasse *Klant*). Uniciteitsregel *c1* in de Kenniskunde informatie grammatica van figuur 7 komt overeen met een aantal attribuutmultipliciteiten, rolmultipliciteiten en beperkingsregels. Voor elke rol uit feittype *Ft2* die *niet* wordt overdekt door de uniciteitsregel *c1*, vinden we een multipliciteit of een beperkingsregel in het corresponderende UML klassediagram in figuur 19. M.b.t. de rol *klantnaam* uit *Ft2* wordt uniciteitsregel *c1* gecodeerd als de waarde 1 voor de maximum attribuut multipliciteit van het attribuut klantnaam van de klasse *Klant.* De rollen *datum*, *rijbewijs, telefoonaansluiting* en *verhuurverleden* van feittype *Ft2* kunnen we eveneens afbeelden als een maximum multipliciteit van 1 voor de bijbehorende klassenattributen. We merken op dat in het geval er geen expliciete attribuut-multipliciteit wordt vermeld, we ervan moeten uitgaan dat deze gelijk is aan 1. De rol *adres* van feittype *Ft2* is gecodeerd in UML als de associatie *woont op*. De beperkingsregel *c1* is voor deze rol geëncodeerd als de maximum multipliciteit van 1 voor de rol in deze associatie gespeeld door de klasse *Adres*.

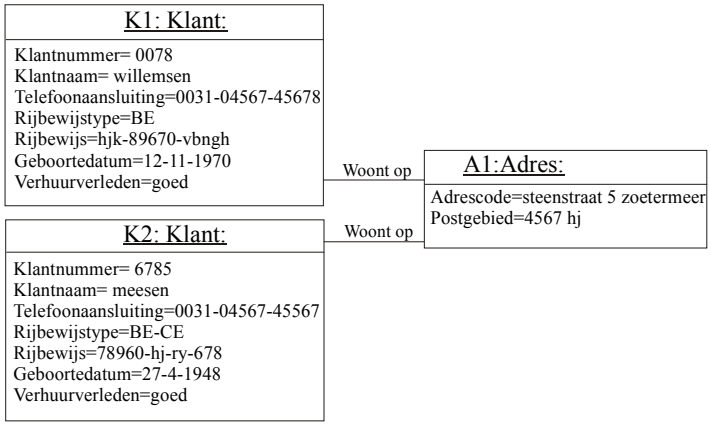

Fig.20: UML objekt diagram

Verplichte rollen die ten alle tijden afgeleid kunnen worden uit andere rollen binnen de applicatie informatiegrammatica kunnen worden gemodelleerd als afgeleide attributen (bijvoorbeeld *verhuurverleden*) of afgeleide associaties in UML. De afleidingsregels voor de rollen die op een willekeurig moment kunnen worden afgeleid kan men modelleren als operatie en de bijbehorende methode van een objekt klasse. Zo wordt afleidingsregel AR1 geïmplementeerd in UML als de operatie *huurovereenkomst bedrag()* behorende tot de klasse *Huurovereenkomst*. Deze operatie wordt aangeroepen in een andere operatie die is gedefinieerd bij deze objekt-klasse, namelijk de operatie *maak reservering*. Afleidingsregels AR2, AR3 en AR4 worden respectievelijk geïmplementeerd binnen de operaties *breng leenauto terug()* van klasse *Huurovereenkomst*, en de operaties *veranderonderhoudstatus()* en *veranderverhuurstatus()* van klasse *Leenauto*. In figuur 19 is het klassendiagram gegeven waarin de applicatie informatiegrammatica uit figuur 11 is afgebeeld tezamen met de verwijzingen naar de afleidingsregels uit de procesbeschrijving van figuren 12 en 13.

In tegenstelling tot de applicatie-informatiegrammatica of regelgeving in Kenniskunde kan men het klassendiagram in UML *niet* populeren. De applicatie informatiebank oftewel de 'zinnen' uit de gebruikersvoorbeelden kan men in UML echter representeren als objektdiagrammen. Deze diagrammen zijn al gauw onoverzichtelijk, omdat de instanties uit de informatiebank op nagenoeg dezelfde wijze worden gerepresenteerd als de elementen uit het klassendiagram. We hebben de feitinstanties die overeenkomen met het voorbeeld in figuur 1 gerepresenteerd in een UML objekt diagram in figuur 20. Voor een uitgebreid overzicht van de manieren waarop NIAM/ORM feittypen en beperkingsregels kunnen worden uitgedrukt in UML verwijzen we naar [3-8].

3.3 De sequence en collaboration diagrammen

In deze paragraaf zullen we aandacht besteden aan de volgende UML diagramsoorten: *sequence* en *collaboration* diagrammen. Deze diagramsoorten worden tezamen ookwel aangeduid als *interactiediagrammen* [1, 14]. Volgens de UML literatuur [1, p.208], representeert een interactie die hoort bij een use case, een scenario die de volgorde van interacties binnen zo'n use case precies voorschrijft. We zagen dat de samenhang tussen de diagramsoorten *use casediagram* en *klassendiagram* in de vorm van klassenoperaties voor de use cases uit het use-case diagram bestond. We zullen nu laten zien dat de samenhang tussen het *collaborationdiagram* (of *sequence diagram*) enerzijds en de *use-case* en *objektdiagrammen* anderzijds wordt gevormd door de (instanties van) *actoren* uit het use-case diagram en de instanties van *objekt-klassen* uit het objekt-diagram.

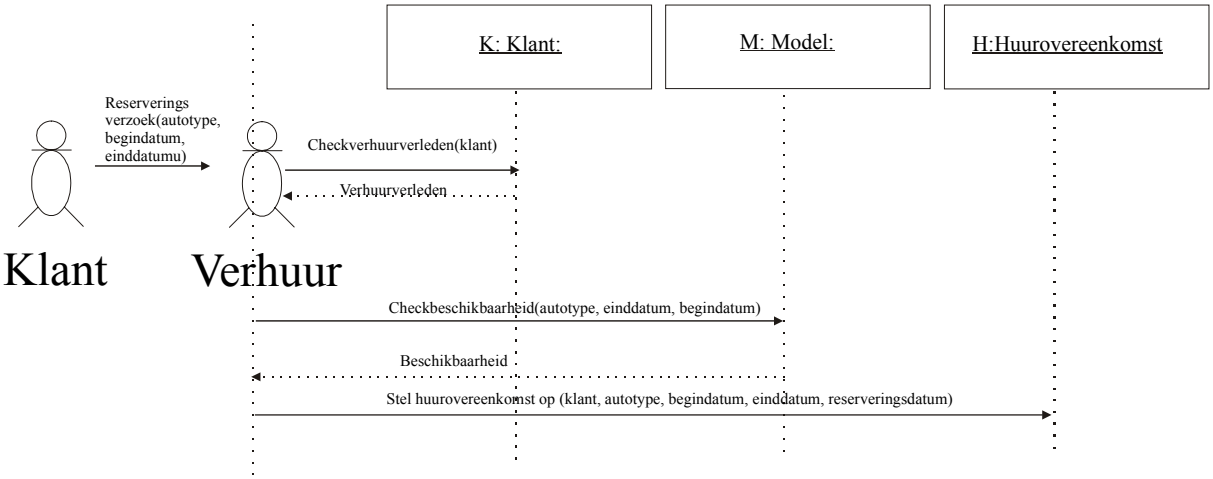

Fig.21: UML sequencediagram voor de use case reserveer auto

Een belangrijk modelleringconcept binnen de objektgeoriënteerde modelleringstalen is het concept van *bericht*. Een bericht dient te worden geadresseerd aan een specifiek objektinstantie. Zogauw deze objektinstantie het bericht heeft ontvangen zal de objektinstantie de bij dit

berichtbehorende operatie executeren, hetzij een waarde terugsturen, hetzij een signaal sturen naar (een of meer andere) objekten. Het ontvangen van een bericht kan worden beschouwd als het zich voordoen van een event [1, p.208]. Als een bericht dient te worden gebruikt voor het aanroepen van een operatie bij het geadresseerde objekt, dan dient zo'n bericht dezelfde naam te hebben als de operatie die moet worden aangeroepen. Onder de mogelijke berichtsoorten bevinden zich aanroepen van operaties voor het creëren, respectievelijk het verwijderen van objekten van een bepaalde objekt klasse[4]. Hier wordt dus de samenhang tussen de interactiediagrammen en het klasse-diagram geïllustreerd. In UML dienen objekten die met elkaar interacteren in een interactiediagram, d.m.v. het uitwisselen van berichten aan elkaar gelinkt te zijn. Dit betekent dat het mogelijk is in UML om (tijdelijke) links te creëren tussen 2 objekten van objektklassen waartussen *geen* associatie is gedefinieerd. In de *sequence*- en *collaboration*diagrammen van UML wordt de precieze samenhang tussen de actoren, de objekten en de berichten die worden uitgewisseld afgebeeld. De precieze instantiatie parameters voor een specifieke operatie-aanroep zijn niet gespecificeerd in een use-case diagram evenals de precieze beschrijving van het gedrag van het applicatiesysteem, het sequence- of collaborationdiagram laat dit wel zien. Als we het sequence-diagram uit figuur 21 bekijken waarin het gedrag van de sub-use case *maak reservering* wordt uitgewerkt, dan kunnen we zien dat er actoren kunnen worden toegevoegd die *niet direct* communiceren met het applicatiesysteem (bijvoorbeeld de actor klant) in zoverre deze berichten uitwisselen met actoren uit de use-case diagrammen van het applicatiegebied (bijvoorbeeld de actor *verhuur*).

Als we een significant deel van het voorbeeld van een sequence diagram in figuur 21 verwoorden krijgen we de volgende zinnen:

Een instantie van de actor klant stuurt een bericht van het type reserveringsverzoek naar een instantie van de actor verhuur.....(zin 1)
Een instantie van de actor verhuuragent stuurt een bericht van het type checkverhuurverleden naar een instantie van de klasse Klant……(zin 2)
Een instantie van de actor verhuuragent stuurt een bericht van het type checkbeschikbaarheid naar een instantie van de klasse Model……(zin3)

Als we nu proberen om vanuit deze significante verzameling zinnen, het verwoorde deel van het voorbeeld te reconstrueren, dan zien we dat impliciete volgorde uit het sequence diagram niet behouden blijft. Een sequence diagram, suggereert dus ten onrechte dat de volgorde van berichtenverkeer word vastgelegd. In figuur 22 hebben we het collaboration diagram weergegeven dat hoort bij het sequence diagram van figuur 21. We merken op dat de volgorde waarin de boodschappen verstuurd worden is terug te vinden in de volgnummers van de interacties.

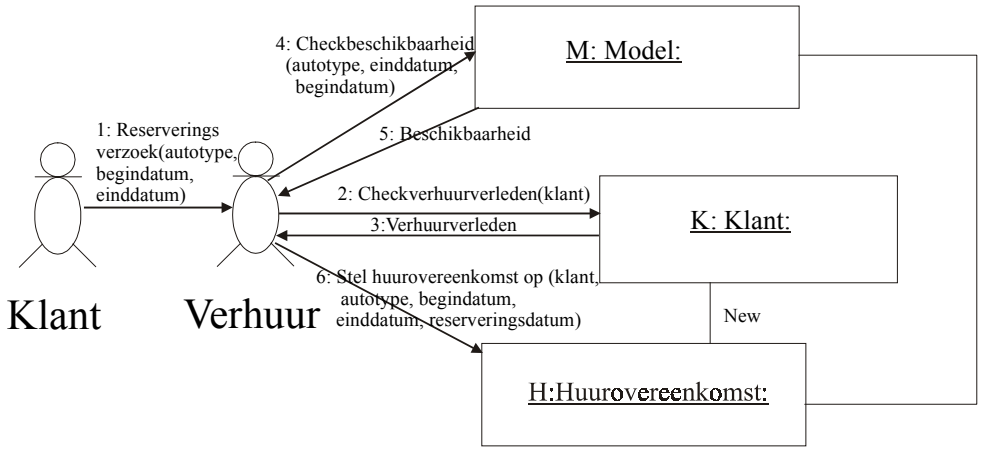

Fig.22: UML Collaboration diagram

[4] Indien we besluiten om objekten te identificeren d.m.v. naamattributen, kan de specifieke objektaanroep gemodelleerd worden als argumentwaarde(n) voor (de) het naamattribu(u)t(en) in het bericht.

De verwoording van een significant deel van het collaboration-diagram uit figuur 22 is de volgende:

Een instantie van de actor klant stuurt een bericht met volgnummer 1 naar een instantie van de actor verhuur……(zin 4)
Een instantie van de actor verhuur stuurt een bericht met volgnummer 2 naar een instantie van de klasse klant……(zin 5)
Een instantie van de actor verhuur stuurt een bericht met volgnummer 4 naar een instantie van de klasse Model……(zin 6)
Het bericht met volgnummer 1 is een instantie van het berichttype reserveringsverzoek……(zin 7)
Het bericht met volgnummer 2 is een instantie van het berichttype checkverhuurverleden……(zin 8)
Het bericht met volgnummer 4 is een instantie van het berichttype checkbeschikbaarheid……(zin 9)

Het is duidelijk dat een collaboratie-diagram méér informatie bevat dan het sequence-diagram, en ze derhalve *niet* equivalent zijn. Het probleem is echter dat een collaboration-diagram geen condities op de volgorde en het versturen van berichten kan bevatten. Zo zal het niet mogelijk zijn om aan te geven in het collaboratie-diagram van figuur 22 dat in het geval een klant voor het eerst zaken doet met Herst auto-verhuur, het verhuurverleden niet gecheckt hoeft te worden. Anderzijds dwingt het collaboratie-diagram tot het specificeren van een volgorde, ook in die gevallen waarin dit niet strict noodzakelijk is. Er is geen reden in ons voorbeeld om af te dwingen dat het verhuurverleden *voor* de beschikbaarheid dient te worden gechekt. De enige restrictie op de volgorde is dat de beschikbaarheid en het verhuurverleden gecheckt dienen te zijn *voordat* een huurovereenkomst wordt opgesteld. Verder kan men ook niet precies aangeven hoe de specifieke waarden die dienen te worden meegegeven aan de parameters van de berichtsoorten met elkaar samenhangen. Als we bijvoorbeeld kijken naar de parameter *begindatum* die hoort bij de berichten 1, 4 en 6 dan kunnen we nergens uit afleiden dat de instantie van deze datum op alle plaatsen dezelfde waarde dient te hebben. Dit betekent dat de interactiediagrammen zowel *over*-specificeren als *onder*-specificeren. De interactiediagrammen kan men vergelijken met GANNT-charts die gebruikt worden voor project-planning. In een complexe en dynamische omgeving waarin planningen frequent moeten worden gewijzigd zijn planningsmethoden als netwerk-planning veruit superieur omdat zijn de noodzakelijke precedentie-relatie tussen activiteiten bevatten. Heeft men eenmaal een nieuwe planning gemaakt met behulp van een netwerk-methodiek, dan kan de GANNT-chart hier direct uit worden afgeleid. Het is echter niet mogelijk om vanuit een GANNT-chart een nieuwe planning te maken die de onderliggende precedentie-relaties volledig respecteert. Interactiediagrammen zijn derhalve niet geschikt voor het precies modelleren van het gedrag in een UvD.

3.4 Het toestandsdiagram.

De interactie diagrammen in paragraaf 3.3 modelleren die objekten die samen werken in een bepaalde use case, bijvoorbeeld bij het maken van een verhuurovereenkomst. Om het gedrag van individuele objekten te modelleren kent UML onder meer de diagramsoort *toestandsdiagram* voor het modelleren van de populatie-overgangsregels in Kenniskunde. In figuur 23 hebben we toestandovergangsregel *c10* uit figuur 9 gemodelleerd als toestandsdiagram met het bijbehorende UML klassendiagram.

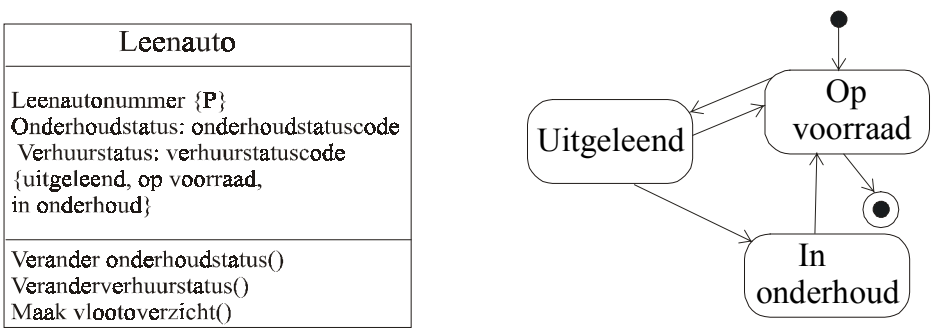

Fig. 23: UML toestandsdiagram van overgangsbeperkingsregel c10 gedefinieerd op klasse leenauto

Teneinde de dynamische domeinaspecten precies te beschrijven kunnen we een uitgebreide vorm van een toestandsdiagram maken waarin we *geavanceerde toestanden* [1] onderkennen, waarin de samenhang tussen events, de toestanden van objekten en de acties die leiden tot het aanroepen van klasse-operaties wordt gespecificeerd. In figuur 24b hebben we de eventregel uit figuur 15 weergegeven als een geavanceerd toestandsdiagram. De eventtypen uit de Kenniskunde modellen worden weergegeven als event-klassen, en derhalve het zich voordoen van een event van zo'n eventtype als de objekt-toestand van de corresponderende klasse in een statechart diagram. De condities tussen de event argumenten en de informatiebank worden gemodelleerd als 'guard' conditie in het geavanceerde toestandsdiagram. Het aanroepen van een proces-type in de Kenniskunde eventregels wordt gemodelleerd als een actie die samenvalt met een transitie in het toestandsdiagram.

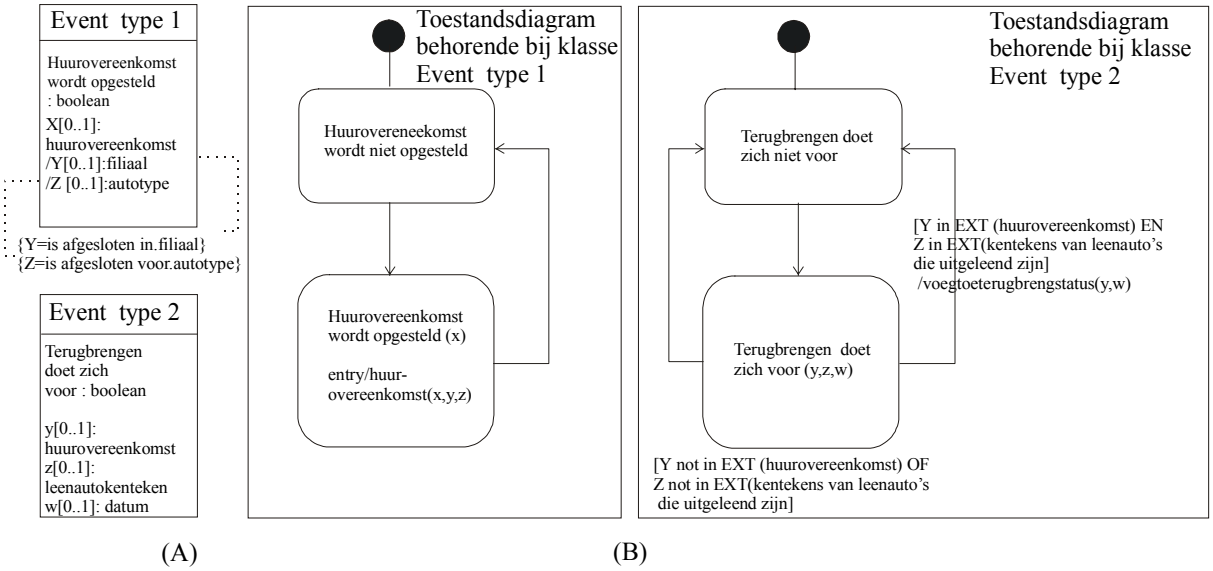

Fig.24: Geavanceerd toestandsdiagram voor de event regel uit figuur 15.

Voor elke actie dienen de volgende zaken gespecificeerd te worden: een actieverzoek, bestaande uit het (de) objekt(en) die de actie dienen uit te voeren, de operatie(s) die dien(en)t te worden geëxecuteerd door de aangeroepen objekten, een lijst van waarden voor de argumenten en het aantal malen dat de betreffende operaties dienen te worden uitgevoerd (recurrence). Verder kan de impuls-parametertoewijzer van de eventregel uit figuur 15 geïmplementeerd worden door het specificeren van de attribuuttypen voor de klasse-attributen van de objekt klasse *event type 2*. Het is verder mogelijk om guard-condities aan uit te voeren acties in een toestand toe te voegen. Op het eerste gezicht lijkt een (geavanceerd) toestandsdiagram bij uitstek geschikt om de eventbeschrijving van ons voorbeeld te modelleren in UML.

De samenhang tussen het (geavanceerde) toestandsdiagram en het klassendiagram wordt gevormd door de akties die zijn gedefinieerd bij het binnenkomen, respectievelijk het verlaten van een toestand. Deze acties kunnen samenvallen met het aanroepen van een of meer klasse-operaties. Verder is er een samenhang met het objektdiagram, in de zin dat een specifieke configuratie van attribuutwaarden van een objekt een specifieke toestand in het toestandsdiagram definieert. Als tweede samenhang met het objektdiagram kan men de argumenten in de (guard)-condities van het toestandsdiagram beschouwen, zij verwijzen naar het al dan niet bestaan van attribuutwaarden en links in het objekt-diagram.

3.5 *Het activiteitendiagram*

De laatste modelsoort die UML kent voor het beschrijven van het gedrag van een willekeurig bedrijfssysteem is het activiteitendiagram. In de definiërende UML literatuur [1] wordt verteld dat een activiteitendiagram kan worden toegevoegd aan elk modelelement teneinde het gedrag van zo'n modelelement te beschrijven. Volgens de UML literatuur is het meest toepasselijke modelelement

wiens gedrag beschreven kan worden door een activiteitendiagram een *operatie*. In figuur 25 hebben we de operatie c.q. de methode *breng leenauto terug()* weergegeven in een activiteitendiagram.

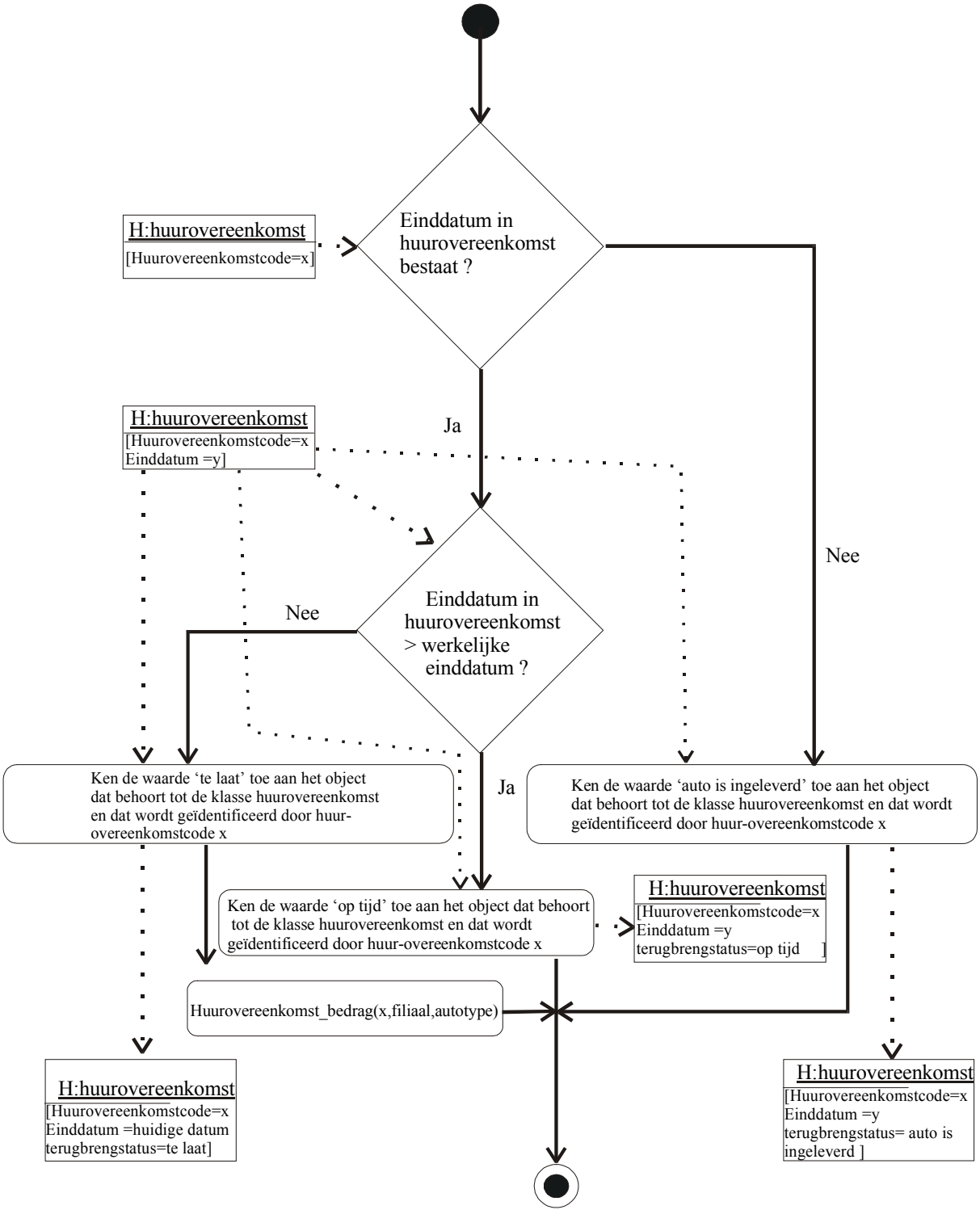

Fig.25: Activiteitendiagram van operatie voeg_toe_terugbrengstatus(huurovereenkomst, datum)

Een aktiviteitendiagram is eigenlijk bedoeld als een soort 'flow-chart'. Het beschrijft wat er precies gebeurt op het moment dat de bijbehorende operatie wordt aangeroepen. Het startmoment van de activiteit wordt weergegeven door een 'donkere' stip, vanaf dat moment kunnen er acties worden uitgevoerd, zoals het berekenen en toekennen van een waarde voor een of meer klasse-attributen of het creëeren van een nieuwe link voor een associatie die is gedefinieerd in het klasse-diagram. Deze acties

kunnen onconditioneel of conditioneel worden uitgevoerd. In het laatste geval is de uitvoering van een bepaalde actie afhankelijk van de attribuutwaarden of linkeinden in het objekt-diagram of van het antwoord van een gebruiker van het bedrijfssysteem (bijvoorbeeld de conditie dat de einddatum in de huurovereenkomst later is dan de huidige datum). De eindtoestand voor een aktiviteit wordt weergegeven d.m.v. twee 'concentrische' cirkels waarvan de binnenste is 'gevuld' (zie figuur 25).

De samenhang tussen het *aktiviteitendiagram* en het *objektdiagram* bestaat hieruit dat in een aktietoestand de attribuutwaarden in een objektdiagram kunnen worden vastgesteld of dat er links in een objecktdiagram kunnen woren gecreëerd als resultaat van een aktiviteit. Een tweede samenhang tussen deze modelsoorten is te vinden bij het toetsen van sommige condities in het aktiviteitendiagram. Hier dient het bijbehorende objektdiagram geinspecteerd te worden. De samenhang tussen het aktiviteitendiagram en het klasse-diagram wordt gevormd door het feit dat sommige actietoestanden overeenkomen met het aanroepen van een of meer operaties uit het klasse-diagram.

3.6 Het componentdiagram

De 3 conceptuele modelsoorten die zijn gedefinieerd in Kenniskunde maken het mogelijk om elke toepassingsdomein m.b.t. de informationele aspecten te modelleren. De 7 diagramsoorten in UML die we tot nu tot hebben behandeld kan men in principe gebruiken voor het beschrijven van de informationele aspecten van een willekeurig toepassingsgebied op conceptueel niveau. Er zijn echter twee modelsoorten in UML die voorbehouden zijn aan specifieke domeinen. De eerste van deze modelsoorten is het *componentdiagram*. In de UML literatuur [1] wordt een componentdiagram gebruikt om een softwareconfiguratie in kaart te brengen. In figuur 26 vinden we een voorbeeld van zo'n componentdiagram.

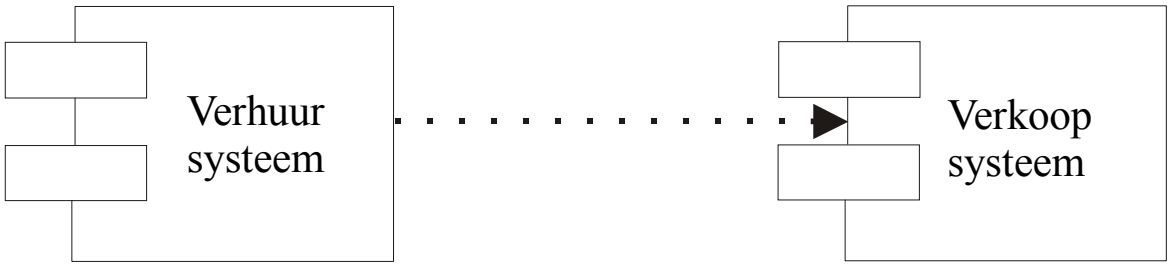

Fig 26: Componentdiagram

3.7 Het deployment diagram

De tweede UML diagramsoort die is voorbehouden aan een specifiek domein is het *deployment-diagram*. Een *deploymentdiagram* laat zien hoe de fysieke objekten in het run-time systeem en de objekten die hierop worden geëxecuteerd verbonden zijn. We kunnen zeggen dat een UML deployment diagram laat zien hoe een computersysteem is geïmplementeerd in termen van processing en communicatie hardware. In figuur 27 hebben we een voorbeeld gegeven van een deployment diagram.

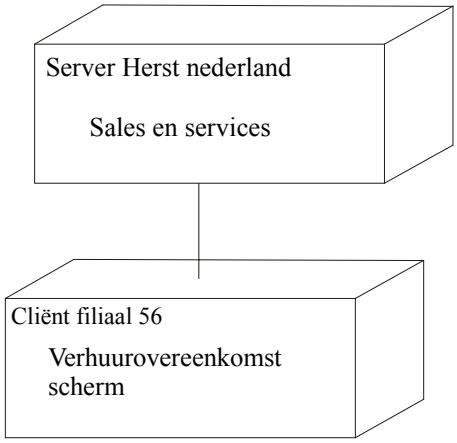

Fig 27: Deployment diagram

4. UML diagramsoorten voor de conceptuele bedrijfsmodellering

We kunnen concluderen dat er geen concrete afbakening in UML bestaat die het mogelijk maakt om te zeggen welke 'concrete' documenten zijn bevat in het universum van discussie. Het verdient derhalve aanbeveling om zogenaamde concrete gebruikersvoorbeelden (ookwel *data use-cases* genoemd [8]) te gebruiken. De inhoud van deze gebruikersvoorbeelden kan worden gemodelleerd op typeniveau in het *klassendiagram* en op instantieniveau in het *objektdiagram*. Een deel van het procesperspektief kan worden afgebeeld als methoden in het klassendiagram een ander deel van dit perspektief wordt afgebeeld in een of meer van de volgende UML modelsoorten: toestandsdiagrammen en activiteitendiagrammen. Tenslotte kunnen we het eventperspektief modelleren in UML in een of meer van de volgende UML modelsoorten; sequencediagrammen, collaborationdiagrammen, uitgebreide toestandsdiagrammen en activiteitendiagrammen. In figuur 28 hebben we de samenhang tussen de modelsoorten en de respectievelijke modelleringconcepten in Kenniskunde en UML samengevat.

UML modelsoort	**Universum van Discussie**	**Bevat de volgende Modelleringsconstructies uit NLM**	**NLM Modelleringsconstructie is bevat in de volgende NLM modelsoort**
Use-case diagram	allen	Namen van processen	Procesbeschrijving
Klassen diagram	allen	feittypen	Applicatie informatiegrammatica
		toestandsbeperkingsregels	Applicatie informatiegrammatica
		Proces-aanroepen	Procesbeschrijving
		afleidingsregels	Procesbeschrijving
		Impulsparameter toewijzer	Eventbeschrijving
		eventtypen	Eventbeschrijving
Objekt diagram	allen	Feitinstanties	Applicatie informatiebank
Sequence diagram	allen	informatiebankprocessen	Procesbeschrijving
Collaboration diagram	allen	informatiebankprocessen	Procesbeschrijving
Toestandsdiagram	allen	toestandovergangsbep.reg	Applicatie informatiegrammatica
Uitgebreid toestandsdiagram.	allen	Events Condities Proces-aanroepen	Eventbeschrijving
Activiteitendiagram	allen	Pre-condities Post-condities afleidingsregels	Procesbeschrijving
Componentdiagram	Software configuratie		
Deploymentdiagram	Fysieke implementatie		

Fig. 28: Correspondentie UML en KENNISKUNDE modelsoorten

In Kenniskunde bestaat er een strikte volgorde waarin de verschillende applicatiegebonden modellen dienen te worden gecreëerd [11]. Allereerst dient het universum van discussie te worden afgebakend door de concrete voorbeelden van communicatie te verzamelen of op te stellen die van belang zijn voor het bedrijfssysteem. Deze voorbeelden oftewel data-use cases worden tezamen het Universum van Discussie genoemd. Vervolgens worden de voorbeelden verwoord door gebruikers. In samenspraak met de informatie-analist worden vervolgens de feittypen, de zinsjablonen, de populatie beperkingsregels en de populatie-overgangsbeperkingsregels afgeleid. Het zo verkregen model wordt de applicatie informatiegrammatica genoemd. Als de applicatie-informatiegrammatica bekend is het relatief makkelijk om te bepalen welke afleidingsregels ertussen feittypen van deze grammatica bestaan. Deze afleidingsregels worden vastgelegd in de procesbeschrijving. Als eenmaal de informatiegrammatica en procesbeschrijving bekend zijn is het relatief eenvoudig vast te stellen welke event typen aanleiding vormen voor het opstarten van processen uit de procesbeschrijving. We kunnen nu een voorschrift geven m.b.t. de volgorde waarin (delen van) de conceptuele UML modellen dienen te worden gemaakt:

1. Selecteer de relevante *data use-cases*.
2. Maak voor elke *data use-cases* ad 1) een *use case*.
3. Maak het initiële *klassendiagram* waarin het informatieperspektief wordt gemodelleerd; de domeinklassen met klasse-attributen (boven de streep) en associaties en multipliciteiten en tekstuele constraints gebaseerd op de *data use-cases* ad 1). Gebruik naamattributen om individuelen objekten aan te duiden.
4. Analyseer het klasse-diagram (bijvoorbeeld naar het patroon van attribuut-multipliciteiten) en definieer indien nodig sub-use cases voor de use cases ad 2)
5. Voeg de *operaties* die samen vallen met (sub) use-cases toe aan de objektklassen uit het *klassendiagram* en kwalificeer de attributen en associaties die afgeleid kunnen worden als *afgeleide* attributen respectievelijk afgeleide associaties. Specificeer eveneens de bijbehorende afleidingsregels als tekstuele (of OCL) constraints.
6. Deklareer *sub-operaties* die onderdeel vormen van twee of meer andere operaties als aparte operaties bij de betreffende *klasse*.
7. Maak een *activiteitendiagram* voor elke *operatie*.
8. Maak een *toestandsdiagram* die een populatie-overgangsbeperking weergeeft voor elke toestandsovergangsbeperkingsregel.
9. Voeg de *eventtype-klassen* toe aan het *klassendiagram* en maak een bijbehorend uitgebreid toestandsdiagram voor elke impuls. Creëer een *afgeleid attribuut* voor elke niet-triviale impulsparameter toewijzer.

Als bovenstaande volgorde wordt aangehouden, wordt het modelleren in UML een stuk gemakkelijker omdat men zich dan kan concentreren op een aantal wezenlijke vragen. De *data use-cases* illustreren eveneens de concrete uitkomst van een *use-case*. Vervolgens kunnen de concepten en entiteiten binnen het applicatiedomein en hun relaties heel eenvoudig in kaart worden gebracht in de vorm van objekt klassen, subtypen, klassenattributen en de associaties tussen objekt klassen. De roleigenschappen en populatiebeperkingsregels kunnen vervolgens worden gemodelleerd als attribuut multipliciteiten, associatie-einde multipliciteiten en tekstuele beperkingsregels in UML. Toestandsdiagrammen kunnen gebruikt worden om de populatie-overgangsregels weer te geven. Vervolgens kan worden gekeken welke van de attributen, respectievelijk associaties, *afgeleide* attributen respectievelijk *afgeleide* associaties zijn. Voor elke operatie kan men dan een activiteitendiagram maken waarin de samenhang tussen objekten, links uit het objektdiagram en het resultaat van de operatie zichtbaar wordt. Als laatste stap kan men impulsklassen definiëren en aangeven onder welke condities het voorkomen van een specifieke impulstoestand al dan niet leidt tot een actie c.q. het aanroepen van een operatie in een objekt. Aangezien UML is gebaseerd op het OO-paradigma, kent UML het concept van *polymorfisme*. Dit betekent dat één operatie op verschillende manieren kan worden geïmplementeerd in verschillende objektklassen. Om duidelijk te maken welke implementatie van een operatie in een specifiek geval dient te worden aangeroepen (methode) en aan welke objekt-instantie het bericht is geadresseerd kan het noodzakelijk zijn om een *interactie diagram* te maken. Indien het vanuit het eerder verkregen klassendiagram duidelijk is dat een specifieke operatie slechts in één objektklasse is

gedefinieerd en er gebruik wordt gemaakt van naam-attributen kan dit echter achterwege blijven. In dit geval kan men het conceptuele schema voor de drie informatiekundige perspektieven in Kenniskunde volledig beschrijven door het maken van de juiste instanties van de volgende UML diagramsoorten:

1. Use case diagram (inclusief sub-use cases)
2. Objekt klasse-diagram (inclusief event-klassen)
3. Aktiviteitendiagram (voor elke operatie)
4. (Geavanceerd) toestandsdiagram (voor elke overgangsbeperkingsregel en impuls)

5. Conclusies

In dit artikel hebben we vanuit de conceptuele modelsoorten die gedefinieerd zijn in de modelleringsmethodologie Kenniskunde, de samenhang tussen de 9 UML modelsoorten gegeven (zie figuur 29). Als we nu Kenniskunde vergelijken met de 9 UML diagramsoorten dan kunnen we concluderen dat het in UML allereerst moeilijk is om alle beprekingsregels op het voorkomen van zinpopulaties te modelleren. Ten eerste omdat het klassendiagram in UML niet populeerbaar is. Ten tweede zijn de UML modelleringsconstructies die beschikbaar zijn voor het afbeelden van beperkingsregels niet orthogonaal. Ten derde dient men veel voorkomende beperkingsregels, die men in Kenniskunde grafisch kan weergeven, veelal als tekstuele beperkingsregels te specificeren in UML.

Het belangrijkste verschil tussen Kenniskunde en UML ligt echter in de taalgebaseerde benadering van Kenniskunde. Zo worden niet alleen semantische relaties tussen concepten gemodelleerd, maar wordt er in Kenniskunde eveneens vastgelegd hoe deze semantische relaties dienen te worden gecommuniceerd. In UML is het praktisch onmogelijk om deze gebruikerscommunicatie in de vorm van zinsjablonen vast te leggen.

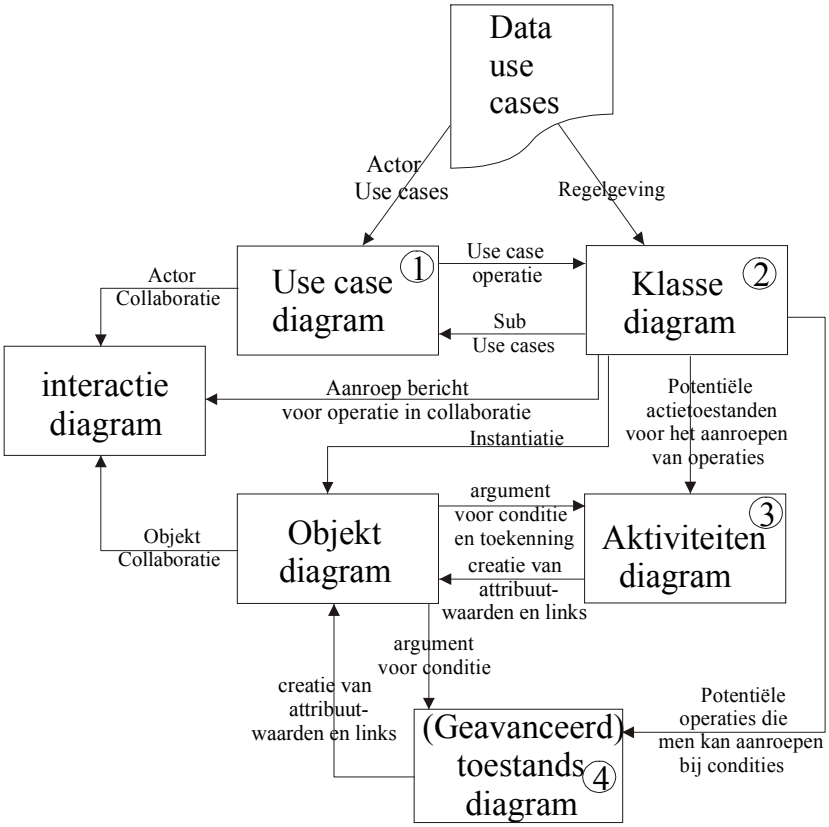

Fig. 29: Samenhang UML diagramsoorten

Een andere tekortkoming van UML is het ontbreken van een modelleringsprocedure. In [11] wordt een precies voorschrift uitgewerkt dat men kan gebruiken om stapsgewijze de Kenniskunde deelmodellen voor het *informatie*, het *proces*- en het *event-perspektief* af te leiden voor een willekeurig

toepassingsgebied. Dit voorschrift bevat tevens kwaliteitschecks voor de analist van alle modelleringstappen. In de huidige UML literatuur is zo'n modelleringsvoorschrift nog niet beschreven. In paragraaf 4 van dit artikel hebben we een aanzet gegeven voor zo'n UML modelleringsvoorschrift.

Literatuur

[1] G.Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide, Addison-Wesley, Reading MA, USA, 1999.

[2] J.Eggink, E. Leenstra, G. Nijssen , Informatie in model: over gegevens, feiten en afspraken, PNA publishing, 1995

[3] T. Halpin, UML data models from an ORM perspective: Part 3, *Journal of Conceptual Modeling*, June 1998.

[4] T. Halpin, A comparison of UML and ORM for data modelling, in: Proceedings EMMSAD'98 3rd IFIP WG8.I International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, 1998.

[5] T. Halpin, Data modeling in UML and ORM revisited, in: Proceedings EMMSAD'99: 4th IFIP WG8.1 International Workshop on Evaluation of Modeling Methods in Systems Analysis and Design, 1999.

[6] T. Halpin, Data modeling in UML and ORM: a comparison, *Journal of Database Management,* 10 (1999).

[7] T. Halpin, Augmenting UML with Fact-orientation , in:workshop proceedings: UML: a critical evaluation and suggested future, HICCS-34 conference, 2001.

[8] T. Halpin, Information Modeling and Relational Databases, Morgan Kaufmann Publishers , 2001.

[9] T. Halpin, Microsoft's new database modelling tool: Part 1, Journal of Conceptual Modeling, June 2001

[10] Microsoft visual studio.net, http://msdn.microsoft.com/vstudio/nextgen/whatsnew.asp, accessed July,26 , 2001

[11] G.Nijssen, Kenniskunde 1A, PNA Publishing, 2002.

[12] G.Nijssen, Universele Informatiekunde, PNA Publishing, 1993.

[13] G.Nijssen, T. Halpin, Conceptual schema and relational database design: a fact oriented approach, Prentice-hall, 1989.

[14] J.Rumbaugh, J Jacobson, G.Booch, The Unified Modeling Language Reference Manual, Addison-Wesley, Reading MA, USA, 1999.

[15] F.Twisk, R. van Montfoort , UI en NIAkM-ISDM: Een concreet alternatief (1), *Informatie* jrg 36 (7/8), p. 438-446, 1994

[16] F.Twisk, R. van Montfoort , UI en NIAM-ISDM: Een concreet alternatiehf (2), *Informatie* jrg 36 (9), p. 512- 518, 1994

[17] *UML version 1.4,* http://www.rational.com/uml/resources/documentation/ summary/index.jtmpl , accessed 12 october, 2002.

[18] J.Wintraecken ,The NIAM Information Analysis Method: Theory and Practice, Kluwer, Deventer, The Netherlands, 1990.