Toegevoegde waarde van kennissystemen

Toegevoegde waarde van kennissystemen

R.V. Schuwer en R.J. Kusters

Een kennissysteem kan worden gekarakteriseerd door een scheiding tussen toepassingsgebied-afhankelijke kennis en toepassingsgebied-onafhankelijke afleidingsregels. Bij gebruik in een bedrijfsomgeving is niet zonder meer duidelijk welke toegevoegde waarde deze scheiding heeft ten opzichte van een conventioneel systeem. Tevens is onduidelijk welke karakteristieken een probleem moet hebben opdat een kennissysteem een goede oplossing is.
Dit artikel tracht antwoorden te geven op deze twee vragen.

1 Inleiding

Veel organisaties hebben in de afgelopen jaren geïnvesteerd in onderzoek naar de toepasbaarheid van kennissystemen voor de oplossing van bedrijfsproblemen. Vaak werden daarbij prototypes ontwikkeld. Het stellen van criteria voor de keuze van het geschikte probleem gebeurde meestal op basis van vuistregels, zoals die bijvoorbeeld zijn te vinden in Waterman (1986): het probleem moet noch te complex, noch te makkelijk zijn; er moet een expert beschikbaar zijn die zijn of haar werkwijze kan en wil verwoorden, enzovoorts. Deze criteria zeggen niets over het kenmerkende van een kennissysteem (namelijk de scheiding tussen kennis en inferentie) en geven ook niet aan wanneer een probleem het best aangepakt kan worden met een kennissysteemoplossing. In dit artikel willen we proberen een antwoord te vinden op deze laatste vraag. Hiervoor is allereerst een precies inzicht nodig in wat een kennissysteem is. In paragraaf 2 schetsen we de ontwikkeling van kennissystemen vanuit de historie als een logische evolutionaire stap. In paragraaf 3 wordt een precieze definitie van een kennissysteem gepresenteerd. De aldus verkregen inzichten dienen als basis voor paragraaf 4 en 5. In paragraaf 4 wordt dieper ingegaan op de toegevoegde waarde van een kennissysteem. Paragraaf 5 vertaalt deze toegevoegde waarde in probleemkarakteristieken, die wijzen op een kennissysteemoplossing. In paragraaf 6 tenslotte staan enkele conclusies.

2 Kennissystemen en achtergronden

In de literatuur zijn vele, meestal informele definities te vinden van een kennissysteem. In dit artikel gaan we uit van de volgende definitie (Mars, 1988):

> Een *kennissysteem* is een computerprogramma, waarin zo goed mogelijk een scheiding is aangebracht tussen toepassingsgebied-onafhankelijke afleidingsregels en toepassingsgebied-specifieke kennis.

Het onderscheid tussen een *kennissysteem* en een *expertsysteem* is gradueel van aard. Onder een expertsysteem wordt vaak een kennissysteem verstaan met prestaties, vergelijkbaar met die van een menselijk expert (zie bijvoorbeeld Waterman, 1986). Zo'n definitie is echter dermate subjectief, dat ze niet als grondslag kan dienen voor een fundamentelere beschouwing. In het vervolg wordt daarom het ruimere begrip 'kennissysteem' gebruikt, zoals hierboven gedefinieerd.

De ontwikkeling van kennissystemen kan worden verklaard vanuit verschillende gezichtspunten. Zo kan het ontstaan van kennissystemen worden verklaard vanuit de wereld van de kunstmatige intelligentie (AI). Uitgangspunt is daarbij het verklaren en nabootsen van menselijk denken en handelen. Met de ontwikkeling van een kennissysteem hoopt men te bereiken dat een computer taken van een menselijke expert kan overnemen en hoopt men tevens meer inzicht te verwerven in het menselijk denken en doen. Voorbeelden van AI-produkten waarbij het menselijk denken en handelen wordt nagebootst, zijn robots, neurale netwerken, computer vision en natuurlijke taalverwerking.

Het ontstaan van kennissystemen kan ook worden verklaard vanuit de trend dat men software steeds meer modulair tracht te construeren. Het doel daarvan is te komen tot hogere kwaliteit en betere onderhoudbaarheid van de software en tot grotere beheersbaarheid van het ontwikkelproces. In dit artikel wordt het ontstaan van kennissystemen met name benaderd vanuit dit gezichtspunt.
De trend naar modulaire constructie van software speelt al vanaf het begin van de automatisering. Van oudsher kan in software de volgende driedeling worden gevonden:
- operating system en utilities (systeemsoftware);
- gegevens;
- applicatieprogrammatuur.

Bij de allereerste computerprogramma's waren alle drie de componenten terug te vinden in één programma. Er was geen sprake van een operating system. Elk programma bevatte laad-, lees- en printfuncties. Als voorbeeld uit deze periode kan worden genoemd de Gamma ET-computer van Bull (1957).

De eerste scheiding in componenten betrof de scheiding tussen systeemsoftware en bewerkingssoftware. Doordat men niet bij ieder programma ook de systeemtaken hoefde te programmeren, werd een grote efficiency-winst bereikt. Hoewel de systeemsoftware aanvankelijk slechts de eerder genoemde functies bevatte, leidde het ontstaan van multiprogramming later tot meer complexe operating systems vanwege de noodzaak van intern geheugenbeheer. Als voorbeeld uit deze periode kan worden genoemd de IBM 1410 (1961).

De tweede scheiding in componenten betrof de scheiding tussen gegevens en applicatieprogrammatuur. Voorbeelden hier zijn onder andere applicaties in de taal COBOL (1961). Toen daarna de mogelijkheid van gelijktijdig ge-

INFORMATIE JAARGANG 32 NR. 12
PAG. 997 T/M 1064 1990

bruik van gegevens door meer gebruikers ontstond, leidde dit tot het afsplitsen van de gegevensbeheersfuncties uit de applicatieprogrammatuur en het ontstaan van een DBMS. Als voorbeeld uit deze periode kan het DBMS IDS (1965) worden genoemd.

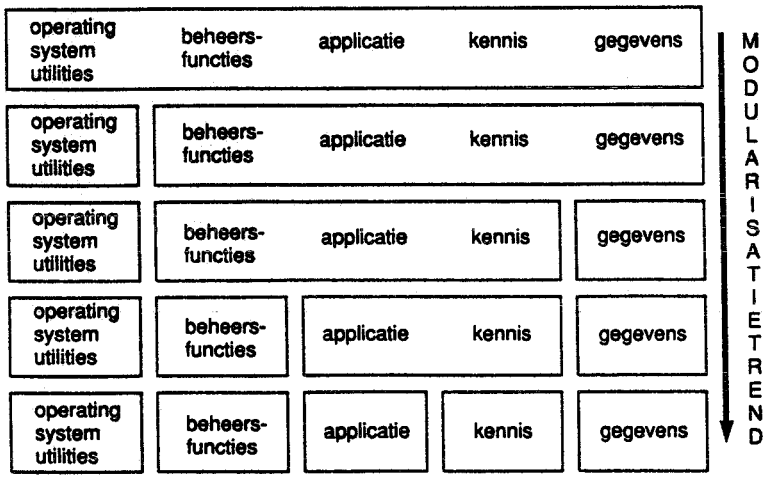
Aansluitend op de scheiding tussen gegevens en applicatieprogrammatuur was de volgende logische stap de afscheiding van kennis uit de applicatieprogrammatuur. Dit resulteerde in het ontstaan van kennissystemen. Hierbij gaat men ervan uit dat een applicatieprogramma twee typen kennis bevat. Enerzijds is dat de domeinafhankelijke kennis over gegevens en over de wijze waarop met gegevens gemanipuleerd kan worden. Anderzijds bevat een applicatieprogramma ook domeinonafhankelijke redeneerkennis. Bij een kennissysteem wordt deze kennis gescheiden van het applicatieprogramma en apart opgeslagen in een kenniscomponent. Deze component bestaat dan uit:

1 domeinafhankelijke kennis over de gegevens;
2 domeinafhankelijke redeneerkennis;
3 domeinonafhankelijke redeneerkennis.

De kennis onder 1 en 2 vormt de toepassingsgebiedspecifieke kennis uit de definitie van Mars. De kennis onder 3 bestaat uit toepassingsgebiedonafhankelijke afleidingsregels. In het applicatieprogramma blijft dan onder andere de user-interface en de programma-control over.

De hier geschetste ontwikkeling is schematisch weergegeven in figuur 1.

In figuur 1 is af te lezen dat 'traditionele' informatiesystemen over het algemeen de volgende structuurcomponenten bevatten, afgezien van harde systeemsoftware:



Figuur 1: Modularisatie van software in de tijd

- gegevenscomponent;
- gegevensbeheerscomponent;
- applicatieprogrammacomponent.

Vergelijking van de componenten van een traditioneel informatiesysteem met die van een kennissysteem levert de volgende verschillen op:

- Het kennissysteem bevat een afzonderlijke kenniscomponent. Deze component maakt bij een traditioneel systeem deel uit van de applicatie-programmacomponent.
- Bij een kennissysteem ontbreekt vooralsnog een component voor het beheersen van de kennis. Ontwikkelingen op dit gebied kunnen leiden tot het ontstaan van zogeheten Knowledge Base Management Systemen (KBMS). In paragraaf 5 gaan we hierop nader in.

Van deze twee verschillen wordt de kenniscomponent als echt kenmerkend voor een kennissysteem beschouwd. Een precieze (formele) beschrijving van een kennissysteem zal derhalve moeten vastleggen hoe zo'n kenniscomponent is opgebouwd. In paragraaf 3 zal hierop verder worden ingegaan.

3 Een formele beschrijving van een kennissysteem

In paragraaf 2 werd een informele definitie van een kennissysteem gegeven. Een nadeel van zo'n definitie is, dat verschillende aspecten van de werking van zo'n systeem onduidelijk blijven:

- Wat zijn precies feiten, regels en waaruit bestaat een inferentiemechanisme?
- Hoe beïnvloeden feiten, regels en inferentiemechanisme elkaar?

Tevens legt zo'n definitie niet vast wat de opbouw van een kenniscomponent is.
Om deze onduidelijkheden op te heffen, is het beter uit te gaan van een formele omschrijving, dus van een formeel model. Het voordeel van zo'n formeel model is, dat eenduidig wordt aangegeven wat een kennissysteem is en hoe de werking van en de samenhang tussen de onderscheiden componenten is. In het kader van dit artikel voert het te ver om dit model uitvoerig toe te lichten. We kiezen daarom voor een sterk vereenvoudigde beschrijving met behulp van een voorbeeld, die uitmondt in een formele definitie van een kennissysteem. Voor een formele opbouw van het model wordt verwezen naar Eiben en Schuwer (1990).

Als voorbeeld kiezen we het volgende:
In het Nederlandse koningshuis zijn talloze relaties aan te geven. Een systeem zal vragen moeten kunnen beantwoorden van de vorm 'Geef alle kleinkinderen van Prins Bern-

INFORMATIE JAARGANG 32 NR. 12
PAG. 997 T/M 1064 1990

hard' of 'Is Willem-Alexander een broer van Johan Friso?' Ter wille van de eenvoud wordt als wortel van de stamboom uitgegaan van het echtpaar 'Hendrik – Wilhelmina'.

Voor de opbouw van een kennissysteem zal, analoog aan een traditioneel systeem, allereerst aangegeven moeten worden, welke relaties in het domein van belang zijn. In het voorbeeld levert dat een verzameling relaties op, waarin onder andere de volgende elementen zijn te vinden:

vader(Persoon,Persoon), neef(Persoon,Persoon),
moeder(Persoon,Persoon), vrouw(Persoon).

Tevens moeten de 'constante' elementen worden aangegeven. In het voorbeeld zijn dat de namen van de diverse leden van de Koninklijke Familie: Hendrik, Wilhelmina, Bernhard, Juliana, enzovoort.
Door relaties te combineren met de constante elementen kan een database-toestand worden gemaakt. Een database-toestand is in feite niet meer dan een verzameling van zulke combinaties. In het vervolg zullen we een element van zo'n database-toestand een *feit* noemen. Een database-toestand wordt ook wel aangeduid met 'feitenbank'. Dit deel van een kennissysteem is vergelijkbaar met de datacomponent (database) van een traditioneel informatiesysteem. Voorbeelden van dergelijke database-toestanden zijn:

u_1 = { vader(hendrik,juliana), moeder(beatrix, johan-friso) }
u_2 = { vader(hendrik,juliana), kind(juliana,hendrik) }
u_3 = { vader(hendrik,juliana) }
u_4 = { vader(hendrik,beatrix), moeder(bernhard, irene), man(bernhard) }

u_1, u_2 en u_3 zijn voorbeelden van database-toestanden die de werkelijkheid op een correcte wijze weergeven. Database-toestand u_2 bevat echter een soort redundantie. Als bekend is, dat vader(hendrik,juliana) geldt, dan 'weet' men, dat kind(juliana,hendrik) ook geldt. Deze onderlinge afhankelijkheden kunnen in de vorm van *regels* worden weergegeven. De algemene vorm van zo'n regel luidt:

Als (een aantal feiten bekend is)
Dan (mag een nieuw feit worden geconcludeerd)

In het geval van u_2 luidt de regel als volgt:

Als vader(Persoon1,Persoon2)
Dan kind(Persoon2,Persoon1).

De verzameling van alle mogelijke regels noemen we een *rulebase*.

Database-toestand u_4 toont aan, dat beperkingen (*constraints*) kunnen gelden voor de combinatiemogelijkheden tussen constanten en relaties. In de database-wereld is dat lang bekend. Zo'n constraint werkt als een filter voor een database. In het vervolg zullen we alleen maar *toegelaten* database-toestanden toestaan. Dit zijn database-toestanden, die aan alle geformuleerde constraints voldoen. De verzameling toegelaten database-toestanden zullen we voortaan het *toegelaten database universum* noemen.

Regels uit de rulebase kunnen worden toegepast op een database-toestand. Aan de database-toestand worden dan nieuwe feiten toegevoegd, zodat dan een nieuwe (toegelaten) toestand wordt verkregen. Dit toepassen van regels uit de rulebase kan herhaald gebeuren, totdat er geen nieuwe feiten meer kunnen worden afgeleid. Een rulebase geeft zo structuur aan het toegelaten database-universum. Deze structuur kan worden beschreven door middel van een functie k. De functie k geeft voor iedere toegelaten database u de uiteindelijk af te leiden toegelaten database-toestand u'. Omdat hierbij gebruik wordt gemaakt van de kennis uit de rulebase, zal de functie k in het vervolg een *kennisfunctie* worden genoemd. Het is te bewijzen dat bij afwezigheid van constraints k (en dus u') eenduidig wordt bepaald door het toegelaten database-universum en de rulebase R.

Samenvattend leidt dit tot de volgende definitie:
Stel dat er een verzameling constraints bestaat en gegeven is.
Een *kennismodel* is dan een tupel $< U,R,k >$, waarbij:
- U het toegelaten database-universum;
- R een verzameling regels (de rulebase);
- k de kennisfunctie, bepaald door U en R.

Een oorspronkelijke database-toestand u uit U vormt tezamen met de rulebase R een *kennisbank*. Een kennisbank is een verzameling feiten en regels. Dit is geheel in overeenstemming met algemeen aanvaarde definities zoals men die vindt in de literatuur (zie bijvoorbeeld Waterman, 1986).

Gegeven een kennismodel $< U,R,k >$ en een kennisbank $< u,R >$ kan een gebruiker vragen stellen aan het systeem. Een vraag (query) bestaat uit een of meer hypothesen. Het doel is nu, dat het systeem nagaat of de hypothese of het tegengestelde daarvan als feit voorkomt in $k(u)$. De verificatie van de hypothesen bestaat uit de volgende stappen:
- Uit de query zal een te verifiëren hypothese worden gekozen. Deze keuze wordt bepaald door een of meer metaregels, de *goal-selectieregels* (deze naamgeving is te verklaren, doordat een te verifiëren hypothese vaak een 'goal' wordt genoemd).

INFORMATIE JAARGANG 32 NR. 12
PAG. 997 T/M 1064 1990

- Als met de beschikbare database de hypothese niet te verifiëren is, zal het systeem net zolang feiten gaan afleiden tot de vraag wel te beantwoorden is òf tot $k(u)$ berekend is. Hiervoor worden een of meer regels uit R geselecteerd, die dan worden toegepast op u. Deze regelselectie wordt bepaald door een of meer metaregels, de *rule-selectieregels.*
- Na het toepassen van de regels uit R is een nieuwe database-toestand u' ontstaan. Ook de verzameling hypothesen G kan veranderd zijn. Zo is het mogelijk dat de oorspronkelijk te verifiëren hypothese vervangen wordt door een aantal nieuwe hypothesen. De verificatie van de nieuwe hypothesen geeft dan tevens antwoord op de oorspronkelijke hypothese. De metaregels die aangeven, welke database-toestand u' en welke verzameling hypothesen G' uit de oorspronkelijke u en G te verkrijgen is, worden *inferentieregels* genoemd.
- Om $k(u)$ te berekenen zullen de juiste inferentieregels in de juiste volgorde moeten worden toegepast. Zo zijn de twee bekende redeneerstrategieën 'forward chaining' en 'backward chaining' te beschrijven door middel van bepaalde combinaties van enkele inferentieregels. De metaregels, die zo'n combinatie kunnen bepalen, worden *stuurvoorschriften* genoemd.
- Als $k(u)$ berekend is, doch de hypothese niet geverifieerd kan worden, zullen een of meer metaregels toch een antwoord moeten geven. Deze metaregels zullen *'onvolledige informatie'-regels* (afgekort OI-regels) worden genoemd.

Samenvattend leidt dit tot de volgende definities:
Een *inferentiemodel* is een tupel $<I,G,S,O>$, waarbij:
- I een verzameling inferentieregels;
- G een verzameling goalselectieregels;
- S een verzameling rule-selectieregels;
- O een verzameling OI-regels.

We kunnen nu een *kennissysteem* als volgt definiëren:
Een kennissysteem is een tupel $<KM,IM,SV>$, waarbij:
- KM een kennismodel;
- IM een inferentiemodel;
- SV een verzameling stuurvoorschriften.

Om het voorgaande te illustreren is in de appendix een voorbeeld uitgewerkt.

Het boven beschreven model geeft eenduidig weer wat onder een kennissysteem wordt verstaan. Het kan tevens worden gebruikt voor het evalueren van hulpmiddelen (talen of shells). Het model is dan leidraad bij het onderzoek hoe problemen worden opgelost bij bijvoorbeeld onvolledige informatie. Dit geeft een indicatie van te verwachten problemen bij het gebruik van een hulpmiddel in een specifieke situatie. Voor een voorbeeld van zo'n onderzoek wordt verwezen naar Eiben en Schuwer (1990).
In het model is de modulaire structuur, zoals in paragraaf 2 werd gepresenteerd, terug te vinden. Zo levert de *KM*-component de gegevens en de domeinafhankelijke kennis over de gegevens. De redeneerkennis wordt weergegeven door de componenten *IM* en *SV*. Hierbij dient te worden opgemerkt dat deze componenten nog verder te modulariseren zijn door onderscheid te maken tussen een domeinafhankelijk en een domeinonafhankelijk deel van de componenten *IM* en *SV*.

Hoewel het gepresenteerde duidelijk maakt, wat precies de kern van een kennissysteem is, geeft het nog geen antwoord op de vraag welke toegevoegde waarde de scheiding tussen kennis en inferentie heeft ten opzichte van een traditioneel informatiesysteem. Tevens wordt niet duidelijk welke karakteristieken een probleem moet hebben, wil een kennisoplossing zinvol zijn. De volgende paragraaf zal ingaan op deze vraagstelling.

4 De toegevoegde waarde van een kennissysteem

In paragraaf 2 en 3 hebben we de modulaire structuur van een kennissysteem als meest kenmerkende eigenschap van dit type systeem gedefinieerd. In deze paragraaf en in paragraaf 5 zullen we deze modulaire structuur als basis gebruiken bij het aangeven van de toegevoegde waarde van een kennissysteem ten opzichte van die van een traditioneel systeem. Hierbij zullen we de terminologie van het formele model uit paragraaf 3 hanteren.

De toegevoegde waarde van een bepaalde vorm van een informatiesysteem kan ontleend worden aan de wijze waarop het systeem voldoet aan de eisen die eraan gesteld worden. In de terminologie van Bemelmans (1987) wordt bij het evalueren van de toegevoegde waarde van een informatiesysteem onderscheid gemaakt in:
- functionele eisen: deze betreffen al die specificaties die aangeven welke gegevens verwerkt en verstrekt moeten worden (*wat* moet het systeem doen);
- niet-functionele eisen (ook wel prestatie-eisen of kwaliteitseisen genoemd): deze betreffen de voorwaarden waaronder gegevensverwerking en -verstrekking zich dienen te voltrekken (*hoe* doet een systeem iets).

De toegevoegde waarde van een kennissysteem kan eveneens vanuit deze invalshoek bekeken worden. Uitgangspunt hierbij zijn de specifieke eigenschappen van een kennissysteem.
Uit de vorige paragraaf is naar voren gekomen dat een kennissysteem wordt gekarakteriseerd door een scheiding tussen *KM* en *IM*. Hiermee is wel aangegeven op welke wijze een kennissysteem is opgebouwd, maar wordt niets gezegd over het soort taken dat kan worden verricht. Er

INFORMATIE JAARGANG 32 NR. 12
PAG. 997 T/M 1064 1990

wordt dus niet aangegeven *wat* een kennissysteem kan doen (functionaliteit), noch op welke wijze, *hoe*, het gedaan kan worden (prestatie).

Door de specifieke architectuur van een kennissysteem zal niet beter aan functionele eisen worden voldaan. In principe kan immers elke functionaliteit die door een kennissysteem geleverd wordt, ook door een traditioneel systeem geleverd worden. Vanuit dat gezichtspunt is het niet verwonderlijk dat een systeem dat in eerste instantie als kennissysteem ontworpen werd, uiteindelijk in een traditionele architectuur geïmplementeerd wordt. Het is dus niet zo dat door het ontwikkelen van een kennissysteem tot nu toe qua functionaliteit onoplosbare problemen kunnen worden aangepakt. Wel zullen bepaalde problemen op eenvoudiger wijze met behulp van een kennissysteem kunnen worden opgelost dan met een traditioneel systeem het geval is. Historisch gezien was het wel zo, dat nog nooit expertise in informatiesystemen werd verwoord. Door de ervaringen die onder andere met de ontwikkeling van kennissystemen werd opgedaan, kon men tot het inzicht komen dat het vastleggen van expertise iets is, dat niet alleen aan kennissystemen is voorbehouden.
De toegevoegde waarde van een kennissysteem zal daarom liggen in het beter kunnen voldoen aan bepaalde niet-functionele eisen.

Bij Bemelmans (1987) wordt een classificatie van niet-functionele eisen gegeven. Van de daar genoemde eisen zijn naar onze mening bij de volgende eisen voordelen te behalen bij een oplossing met behulp van een kennissysteem:

- Grote *complexiteit*: bij een toename van de complexiteit groeit de behoefte naar meer overzicht over het systeem. De verdergaande scheiding in componenten die in een kennissysteem terug te vinden is, zal in principe een dergelijk overzicht bevorderen.
- De *duurzaamheid* van het systeem wordt bevorderd door de scheiding tussen kennis en inferentie. De expliciet onderscheiden inferentie geeft de mogelijkheid tot een flexibeler behandeling van de kennis dan met een 'gewoon' informatiesysteem mogelijk is. Het apart onderkennen en opslaan van de toepassingsgebied-afhankelijke kennis maakt het mogelijk op relatief eenvoudige wijze veranderingen in deze kennis aan te brengen en uitbreidingen toe te voegen.
- Het *inzicht* in de kennis van het systeem is groter. De kennis zit niet meer 'verstopt' in de applicatie, maar wordt expliciet in een kennismodel *KM* weergegeven.

De genoemde voordelen vergroten alle de effectiviteit en efficiëntie van het beheer van de in het systeem aanwezige kennis.

Dat het niet-functionele eisen zijn die uiteindelijk de keuze voor een bepaald type oplossing bepalen, is niet nieuw. Zo worden ook bij traditionele systemen bijvoorbeeld bij de keuze van een vierde- in plaats van een derde-generatietaal, argumenten gebruikt als 'doelmatiger ontwikkeling' en 'betere aanpasbaarheid'. Het is dan niet zo, dat een 3GL de functionaliteit niet zou kunnen leveren, maar het is het gemak, waarmee aan enkele prestatie-eisen kan worden voldaan, dat uiteindelijk de keuze voor een 4GL bepaalt.

Naast genoemde voordelen, die direct zijn terug te voeren op de architectuur van een kennissysteem, is nog een additioneel voordeel te noemen in analogie met ontwikkelingen in de database-wereld. Toen de eerste database-systemen ontstonden, ontstond de behoefte om de gegevens te kunnen beschrijven. Dergelijke beschrijvingen waren in eerste instantie techniekgebonden. Na verloop van tijd ontstond bijvoorbeeld de drie-schema-architectuur, waarbij de beschrijving van gegevens over drie niveaus verdeeld werd.
Op het gebied van kennismodellering is een soortgelijke ontwikkeling waar te nemen. Voor het ontwerp van kennissystemen is het noodzakelijk dat de benodigde kennis in kaart wordt gebracht. Dit gebeurt nu nog hoofdzakelijk in techniekgebonden termen. Er zijn indicaties die wijzen in de richting van een 'conceptueel kennismodel'. De ontwikkelmethode KADS (Breuker en Wielinga, 1988) is een duidelijke aanzet in deze richting. Tijdens de eerste fasen van KADS wordt een conceptueel kennismodel opgesteld in termen, die onafhankelijk zijn van techniek.

Dit 'kennisdenken' hoeft niet beperkt te blijven tot alleen kennissystemen. Ook traditionele systemen kunnen voordelen opdoen met zo'n aanpak. Onder andere wordt het inzicht in de gebruikte kennis groter, doordat deze kennis expliciet gedocumenteerd kan worden. Dit vergroot het inzicht in het totale systeem. De onderhoudbaarheid van het systeem zal hierdoor worden bevorderd.

Hierboven is een aantal punten opgesomd, waarbij de specifieke architectuur van een kennissysteem voordelen oplevert. Voor sommige typen problemen zullen de genoemde niet-functionele eisen een belangrijke rol spelen. In de volgende paragraaf zal dat worden uitgewerkt.

5 Herkennen van probleemkarakteristieken

In de vorige paragraaf werd betoogd, dat de toegevoegde waarde van een kennissysteemoplossing ligt in de mogelijkheid tot het efficiënt en effectief beheer van kennis. Een aanwijzing voor het daadwerkelijk zinvol zijn van een kennissysteem kan worden verkregen door de bij een probleem benodigde kennis te analyseren. Dit idee wordt hierna verder uitgewerkt.

INFORMATIE JAARGANG 32 NR. 12
PAG. 997 T/M 1064 1990

In paragraaf 3 is aangegeven dat de kennis, benodigd voor het oplossen van een probleem, weergegeven kan worden in een kennisbank. De voorgestelde analyse zal zich dus richten op de onderdelen van deze kennisbank, namelijk op de verzamelingen u, waarbij u een element is van het toegelaten database-universum U, en R, de rulebase. Bekeken zal worden, welke eigenschappen van elk van deze verzamelingen richtinggevend zijn bij de beslissing voor het al dan niet gebruiken van een kennissysteemoplossing.

We zullen ons eerst concentreren op de verzameling u. In deze verzameling zijn de beschikbare feiten en de verbanden tussen deze feiten weergegeven (gegevens en gegevensstructuur). Als er sprake is van een verzameling gegevens waarbij hoge eisen worden gesteld aan het beheer van die gegevens, maakt men meestal gebruik van een Data Base Management Systeem (DBMS). Het gebruik van zo'n DBMS wordt gerechtvaardigd door de volgende eigenschappen (Everest, 1986):

- gegevens worden door meer gebruikers en meer applicaties (tegelijk) gebruikt;
- de omvang van de verzameling gegevens is groot;
- wijzigingen in gegevens komen relatief vaak voor.

Bij de rulebase R moeten we eveneens zoeken naar eigenschappen van R, die het wenselijk maken dat er mogelijkheden voor het beheer van R aanwezig zijn. In het algemeen kan men stellen dat het wenselijk is over beheersmogelijkheden te beschikken indien er sprake is van een complexe situatie, van een zich wijzigende situatie of van een situatie waarbij verschillende partijen betrokken zijn. Opgemerkt kan worden dat dit laatste punt geen eigenschap van de situatie is, maar van de wijze waarop met een bepaalde situatie wordt omgegaan. Verder kan worden opgemerkt, dat bij de eigenschappen van het database-universum U deze kenmerken ook naar voren kwamen.

We zullen deze algemene kenmerken vertalen naar kenmerken die betekenis hebben binnen het kader van een rulebase R. Dit levert de volgende eigenschappen van R op:

- De complexiteit en/of de omvang van de verzameling kennis is groot. Naarmate de complexiteit/omvang van een rulebase R toeneemt, zal ook de behoefte ontstaan een beter overzicht over deze rulebase te verkrijgen. Een dergelijk overzicht wordt bevorderd door de architectuur van een kennissysteem. Indien de kennis apart is opgeslagen, is het eenvoudiger hierover een overzicht te krijgen dan wanneer de kennis verborgen zit in een grote hoeveelheid code.
- Indien wijzigingen in de kennis relatief vaak voorkomen, dus de kennis niet robuust is, zijn deze wijzigingen eenvoudiger aan te brengen als de kennis apart is opgeslagen. Men kan dan voorkómen dat delen van de applicatie moeten worden herschreven bij iedere wijziging. Een specifiek geval hiervan is dat een kennisbank nog in ontwikkeling is. De kennis kan dan als onvolledig worden beschouwd.

Zoals reeds opgemerkt is, is ook de wijze waarop gebruik kan worden gemaakt van de verzameling R van belang. Hierbij spelen twee aspecten een rol:

- De kennis wordt door meer gebruikers en meer applicaties (tegelijk) gebruikt. Dit is een situatie die op dit moment weinig voorkomt. Indien meer gebruikers en/of applicaties van dezelfde kennisbank gebruik maken, spelen soortgelijke beveiligingsproblemen als bij een DBMS. Men kan zich bijvoorbeeld voorstellen dat als gevolg van het hanteren van bepaalde regels uit R, feiten afgeleid worden die voor één gebruiker zijn bedoeld, en die afgeschermd moeten worden voor andere gebruikers. In een dergelijke situatie ontstaat de behoefte aan toegangsbeheer tot de kennisbank.
- Een andere relevante eigenschap van de verzameling kennis kan zijn dat de volgorde van kennisregels afhangt van de specifieke toestand van u. Het kan zijn dat in een bepaalde toestand u regel a voor regel b gebruikt dient te worden, terwijl dit in toestand u' precies andersom is.
  Een voorbeeld hiervan is het volgende. Indien er weinig gegevens in toestand u beschikbaar zijn en er relatief veel regels zijn, is het verstandig voor een forward chaining-strategie te kiezen. Bij veel gegevens en weinig regels is backward chaining efficiënter. De keuze van te gebruiken regels en de volgorde van de regels is dan afhankelijk van de inhoud van u. Dit flexibele gebruik van de regels wordt bevorderd door het apart opslaan van kennis in een kennisbank.

We hebben nu zowel voor de verzameling u als voor de verzameling R eigenschappen beschreven die een effectiever en efficiënter beheer van de verzamelingen wenselijk maken. Zoals in de vorige paragraaf werd aangegeven, wordt zo'n mogelijkheid geboden door een kennissysteemoplossing. Dit leidt tot het onderscheiden van de volgende situaties:

- *Situatie I:*
  Zowel bij u als bij R ontbreekt de noodzaak voor een extra inspanning, gericht op een beter beheer. In deze situatie komen kennissystemen niet in aanmerking als oplossing.
- *Situatie II:*
  Er bestaat behoefte aan beheer van gegevens, maar nauwelijks van de regels in R. In deze situatie komt een DBMS in aanmerking als oplossing.
- *Situatie III:*
  Er bestaat behoefte aan het beheer van R, maar niet van de gegevens in u. Dit is een situatie die opgelost kan worden met de huidige generatie kennissystemen. Im-

mers, deze systemen bieden voldoende mogelijkheden voor het beheer van kennis, maar schieten vaak te kort als het op het beheer van gegevens aankomt.

- *Situatie IV:*
  Indien zowel de verzameling gegevens als de verzameling regels van dien aard zijn dat extra beheersmogelijkheden nodig zijn, dan voldoen zowel de huidige generatie kennissystemen (onvoldoende mogelijkheden voor het beheer van u) als de huidige DBMS'en (onvoldoende mogelijkheden voor het beheer van R) niet. In deze situatie ontstaat de behoefte aan een Knowledge Base Management System (KBMS).

Het voorgaande is nog eens samengevat in tabel 1.

R \ u	zwak	sterk
zwak	niet DBMS/KBMS (situatie 1)	DBMS (situatie 2)
sterk	Kennissysteem (situatie 3)	KBMS (situatie 4)

Tabel 1: Overzicht van mogelijke oplossingen, gegeven de eigenschappen van het probleemveld

In een KBMS moeten, behalve de componenten *KM* en *IM* ook functies aanwezig zijn, waarmee u, de rulebase R en (eventueel) de *I*-, *G*-, *S*- en *O*-componenten kunnen worden onderhouden. Dit brengt ons tot een definitie van een KBMS:

Een KBMS is een geheel aan computerprogramma's, waarmee
- **de componenten van een kennissysteem kunnen worden gedefinieerd;**
- **de database (feitenbank) kan worden onderhouden;**
- **de rulebase kan worden onderhouden;**
- **de verzamelingen *I*, *G*, *S* en *O* kunnen worden onderhouden;**
- **de beveiliging van deze componenten kan worden beheerd.**

Merk op dat de functionaliteit van een KBMS zowel die van een DBMS als die van een 'traditioneel' kennissysteem omvat. Als een voorbeeld van een implementatie van een KBMS kan het systeem, zoals beschreven bij Van Herwijnen e.a. (1990) worden beschouwd.

6 Conclusies

Het ontstaan van kennissystemen kan worden gezien als een logische stap in een historische ontwikkeling. Om te bepalen of een probleem met behulp van een kennissysteem adequaat kan worden opgelost, moet kennis die gebruikt wordt bij het oplossen van zo'n probleem bekeken worden. Deze kennis kan worden weergegeven in een verzameling gegevens (u) en een verzameling regels (R). Aan de hand van eigenschappen van deze verzamelingen als grootte, complexiteit, volledigheid en robuustheid en volgorde van gebruik van de regels kan worden aangegeven welk implementatiemechanisme in welke situatie de meest aangewezen oplossing is. Het belangrijkste argument dat bij deze keuze gehanteerd wordt, is de gewenste beheersbaarheid van de verzamelingen u en R.

Appendix

Om de inhoud van het formele model van een kennissysteem te demonstreren volgt hier een voorbeeld. Hierbij worden een kennismodel en een inferentiemodel geformuleerd en wordt de werking van het kennissysteem bij de gegeven inhoud van het inferentiemodel gedemonstreerd.

Gegeven is de volgende kennisbank $<u,R>$:

u = vader(bernhard,beatrix), moeder(beatrix,johan-friso),
getrouwd(bernhard,juliana), getrouwd(claus, beatrix),
moeder(juliana,margriet), vader(hendrik,juliana),
vrouw(juliana), vrouw(beatrix), vrouw(margriet),
not(moeder(juliana,willem-alexander)) }

R = grootvader(X,Y) ← vader(X,Z), moeder(Z,Y). (1)
grootvader(X,Y) ← vader(X,Z), vader(Z,Y). (2)
zuster(X,Y) ← moeder(Z,X), moeder(Z,Y),
vrouw(X), X ≠ Y. (3)
zuster(X,Y) ← vader(Z,X), vader(Z,Y),
vrouw(X), X ≠ Y. (4)
vader(X,Y) ← moeder(Z,Y), getrouwd(X,Z) } (5)

De verzameling te beantwoorden hypothesen is:

H = { grootvader(bernhard,johan-friso), (1)
moeder(beatrix,willem-alexander) } (2)

Het inferentiemodel dat voor deze situatie van toepassing is, luidt als volgt:

G = { 'Selecteer de hypothesen volgens de tekstuele volgorde' }

S = { 'Selecteer de regels waar de te beantwoorden hypothese in het conclusiedeel staat volgens de tekstuele volgorde' }

I = { 'Als de te beantwoorden hypothese als feit in de database voorkomt, dan verwijder de te beantwoorden hypothese uit H (1)'
'Als het tegengestelde van de te beantwoorden hypothese als feit in de database voorkomt, dan ver-

wijder de te beantwoorden hypothese uit H (2).'
'Selecteer een regel volgens een metaregel uit S. Voeg de beweringen uit het conditiedeel van de regel die niet als feit voorkomen in de database toe vooraan H (3)'
'Als alle beweringen uit het conditiedeel van de geselecteerde regel in de database voorkomen, dan voeg de bewering uit het conclusiedeel als feit toe aan de database (4)'
'Als er variabelen voorkomen in de te beantwoorden hypothese, vervang dan de variabelen door constanten door overeenkomstige feiten uit de database te substitueren (5)' 'Bereken $k(u)$' (6)
'Als $k(u)$ berekend is en geen van de inferentieregels (1) t/m (5) kan meer worden toegepast, dan gebruik de OI-regel' (7) }

O = { 'Geef als antwoord FALSE en verwijder de te beantwoorden hypothese uit H' }

SV= { 'Gebruik de inferentieregels in de tekstuele volgorde' }

Merk op, dat de OI-regel in deze situatie overeenkomt met de Closed World Assumption.

Beantwoording van de in H geformuleerde hypothesen gaat als volgt:

1 goalselectie: hypothese 1
2 inferentie: I-regel 3
(opmerking: de I-regels 1 en 2 zijn niet van toepassing.)
3 regelselectie: regel 1
(opmerking: aan H wordt toegevoegd: vader(bernhard,Z) (3) en Moeder(Z,johan-friso) (4).)
4 goalselectie: hypothese 3
(opmerking: aangezien de nieuwe hypothesen links in H zijn toegevoegd, is dit nu de eerste hypothese.)
5 inferentie: I-regel 5
(opmerking: feit 1 geeft: Z = beatrix.)
6 inferentie: I-regel 1
(opmerking: hypothese 3 is nu uit H verwijderd en als feit toegevoegd aan u,
hypothese 4 is nu: moeder(beatrix,johan-friso),
regel 1 is nu: grootvader(bernhard,johan-friso) ← vader(bernhard,beatrix), moeder(beatrix,johan-friso))
7 goalselectie: hypothese 4
8 inferentie: I-regel 1
(opmerking: hypothese 4 is nu uit H verwijderd en als feit toegevoegd aan u.)
9 goalselectie: hypothese 1
10 inferentie: I-regel 3
11 regelselectie: regel 4
12 inferentie: I-regel 4
(opmerking: grootvader(bernhard, johan-friso) wordt als feit toegevoegd aan u.)
13 inferentie: I-regel 1
(opmerking: de hypothese 1 wordt uit H verwijderd, user interface geeft aan de gebruiker de waarde 'TRUE' door.)
14 goalselectie: hypothese 4
15 inferentie: I-regel 6
(opmerking: de I-regels 1-5 zijn niet van toepassing, toegevoegde feiten aan u zijn:
- grootvader(hendrik, margriet),
- zuster(beatrix, margriet),
- vader(claus, johan-friso))
16 inferentie: I-regel 7
(opmerking: geen enkele I-regel is nog toepasbaar, terwijl $k(u)$ al bekend is. Dit betekent dat de OI-regel wordt toegepast.)
17 onvolledige informatie: OI-regel
(opmerking: hypothese 2 wordt uit H verwijderd en de user interface geeft de waarde 'FALSE' door aan de gebruiker.)

Noot

De auteurs zijn dank verschuldigd aan prof. dr. T.M.A. Bemelmans voor zijn kritische opmerkingen bij eerdere versies van dit artikel.

Literatuur

Bemelmans, T.M.A. (1987), *Bestuurlijke informatiesystemen en automatisering*, Stenfert Kroese, Leiden.
Breuker, J.A. en B. Wielinga (1988), 'KADS: een overzicht van een methodologie voor het bouwen van een expertsysteem', *Proceedings NAIC-88*, Amsterdam.
Eiben, A.E. en R.V. Schuwer (1990), 'Kennissystemen: een formeel model', *Proceedings NAIC-90*, Kerkrade.
Everest, G.C. (1986), *Database Management*, McGraw-Hill Book Company, New York.
Herwijnen, J. van, E.G. van Houten, M.A.W. Houtsma en H.M. Romkema (1990), 'Implementatie van een regel-gebaseerd kennissysteem in een relationele database-omgeving', *Informatie* 32-1, blz. 14-21.
Waterman, D.A. (1986), *A guide to expert systems*, Addison-Wesley Publishing Company, Reading (Mass.).

Dr R.J. Kusters heeft bedrijfseconometrie gestudeerd aan de Katholieke Universiteit Brabant. Hij is als universitair docent verbonden aan de vakgroep Bestuurlijke Informatiesystemen en Automatisering van de faculteit Bedrijfskunde van de Technische Universiteit Eindhoven.
Ir. drs. R.V. Schuwer studeerde wiskunde aan de KU te Nijmegen en informatica aan de TU te Eindhoven. Achtereenvolgens was hij werkzaam als docent wiskunde in het middelbaar onderwijs en als consultant van het informatiecentrum binnen Océ-van der Grinten te Venlo. Momenteel is hij werkzaam als universitair docent aan de TU Eindhoven binnen de vakgroep Bestuurlijke Informatiesystemen en Automatisering.