

# PSM: Datamodelleren in het Kwadraat

A.H.M. ter Hofstede, H.A. Proper, Th.P. van der Weide  
E.Proper@acm.org

June 23, 2004

PUBLISHED AS:

A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. PSM: Datamodelleren in het Kwadraat. *DB/Magazine*, 3(4):37–41, June 1992. In Dutch.

## Abstract

De huidige generatie van modelleringstechnieken schieten tekort voor het modelleren van complexe applicatiedomeinen. Voorbeelden van zulke applicatiedomeinen zijn: hypermedia-toepassingen, CAD/CAM systemen, en meta-modellering. Er is een stijgende behoefte aan modelleringstechnieken die het modelleren van dergelijke applicatiedomeinen ondersteunen.

In dit artikel presenteren we PSM, een exponent van een nieuwe generatie van datamodelleringstechnieken, welke geschikt is voor het modelleren van complexe gegevensstructuren.

## 1 Inleiding

De laatste decennia zijn we gewend geraakt aan het steeds maar weer sneller én goedkoper worden van processoren in computers. Daar stond tegenover dat de capaciteit van de opslagmedia niet dramatisch veranderde gedurende lange tijd. Pas de laatste jaren komt hier verandering in. Eerst dienden de optische media zich aan (zoals CD-Rom), en dreigden de magnetische opslagmedia volkomen te gaan overvleugelen. De reactie uit de hoek van de magnetische media was echter verbluffend. Een state-of-the-art personal computer bezit naast een krachtige 80386 processor een snelle harde schijf van zo'n 120 Mb, en zal in de toekomst standaard zijn uitgerust met een CD-Rom drive van 660 Mb.

Een dergelijke computer biedt interessante mogelijkheden. Niet alleen kan men er zeer snel iets op verwerken, men heeft nu ook iets om te verwerken, in de vorm van serieuze hoeveelheden informatie (660 Mb!). Hierdoor ontstaat ruimte voor een geheel nieuw soort van toepassingen, die de mogelijkheden van een conventioneel informatiesysteem (zoals financiële administratie, voorraadadministratie of andere database toepassing) ver overtreffen. Hierbij kunnen we denken aan multimedia-databasesystemen, waarin niet alleen omgegaan kan worden met gewone relationele tabellen, maar ook met teksten, video en geluid.

De informatie die in conventionele systemen ligt opgeslagen betreft volgens schattingen slechts een zeer klein deel van de informatie die in een organisatie rondgaat. Momenteel wordt door velen erkend dat zo'n 95% van alle informatie die omgaat in een organisatie opgeslagen ligt in *documenten* ([Sch89]). Als deze documenten überhaupt elektronisch zijn opgeslagen, blijft het toch meestal bij een (logisch) ongestructureerde wijze van opslaan. Hierbij moet documenten iets ruimer opgevat worden dan normaal gebruikelijk is. Wij zullen in deze context een document beschouwen als een multimedia-document. Een document kan dus ook plaatjes, video en audio bevatten. Typische voorbeelden hiervan zijn WordPerfect documenten, of een video presentatie. Hierin is de enige logische structuur de layout is die de schrijver of regisseur er zelf in heeft aangebracht. Hier zien we een serieus bezwaar van de WYSIWYG aanpak (What You See Is What You Get). Van een document wordt alleen de *layout* vastgelegd. Om interessante vragen te kunnen stellen moeten we echter kennis hebben van de *structuur* (de opbouw) van het document!

Het probleem is dus hoe de informatie in een document moet worden benaderd om bruikbaar te zijn binnen een informatiesysteem. Op de eerste plaats moeten we dan zoeken naar technieken om documenten gestructureerd op te slaan. Vervolgens moet extra aandacht besteed worden aan het bevragen van aldus

gestructureerde gegevensverzamelingen. Het doel van dit verhaal is de database-technieken dusdanig uit te breiden dat ze bruikbaar zijn als structureringsmethode voor *alle* soorten van informatie die rondgaat in een organisatie. Het *bloed* in de *aderen* van de organisatie!

Deze betere opslag en ontsluiting van een grote hoeveelheid informatie is van strategisch belang. Ze zal de informatiestroom doen versnellen, en op basis van deze betere ‘doorbloeding’ tot een gezondere organisatie leiden. Daarnaast ontstaan nu mogelijkheden om dynamische verbindingen te leggen. Je kunt je voorstellen dat je in een jaarverslag een koppeling maakt tussen een diagram van de verkoop over het afgelopen jaar, en de informatie zoals die is opgeslagen in de traditionele relationele tabellen van de verkoop administratie.

In de rest van dit artikel, zullen wij een integratie tussen de wereld der (conventionele) informatiesystemen, en de multimedia wereld nader toelichten. Bij deze integratie wordt het beste van deze twee werelden verkregen (zie figuur 1). Hierbij wordt de 95% informatie die in multimedia-documenten ligt opgeslagen gecombineerd met de informatie die in de conventionele relationele tabellen ligt opgeslagen. Door het toepassen van deze nieuwe mogelijkheden in een organisatie, zal het inzicht in de informatie die door deze organisatie stroomt ver uitgaan boven hetgeen nu mogelijk is. We beginnen daarom in sectie 2 met een overzicht van de PSM datamodellerings-techniek. In sectie 3 richten we ons op een multimedia toepassing.

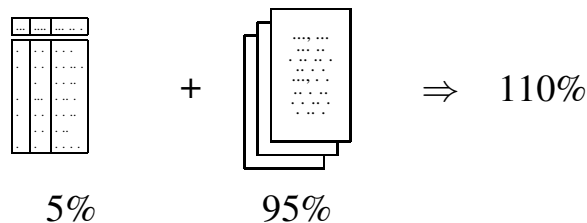


Figure 1: Multimedia en Informatiesystemen: het beste van twee werelden

## 2 De PSM datamodellerings-techniek

In deze sectie introduceren we de datamodellerings-techniek PSM (Predicator Set Model). PSM is een algemene datamodellerings-techniek, die dient als algemene basis voor alle modellerings-technieken die gebaseerd zijn op de zogenaamde *object-rol benadering* ([HW93]). De belangrijkste voorbeelden hiervan zijn ER en NIAM. Dit maakt PSM tot een grootste gemene deler van dergelijke modellerings-technieken. Voordelen van de overgang op PSM zijn daarom:

**Behoud van investering** Alle eventuele bestaande datamodellen kunnen als PSM-model gezien worden. Dit betekent dus het behoud van investeringen.

**Ongewijzigde werkwijze** De werkwijze die samenhangt met datamodellerings-technieken kan men blijven gebruiken. Het enige nadeel is dat men (nog) niet ten volle van alle mogelijkheden die PSM biedt, gebruik kan maken.

**Geen herscholing** Er is niet direct een herscholing nodig van systeemanalisten.

**Snellere ontwerp fase** Door de krachtige modelleringsconcepten van PSM kan overspecificatie vermeden worden, zodat de systeemanalist niet gedwongen wordt zich bezig te houden met niet terzake doende details. Hierdoor kan de systeemanalist zich beperken tot de kern, hetgeen de kwaliteit van het ontwerp ten goede zal komen en het ontwerpproces versnellen.

**Specificaties overzichtelijker** PSM biedt de mogelijkheid tot schemadecompositie. Dit is het opsplitsen van grote schema's in hanteerbare brokstukken. Daardoor kan de systeemanalist zich op een globaal niveau met het schema bezig houden, en zich, waar nodig, richten op de details. Hierdoor is het veel gemakkelijker het overzicht te behouden.

We zullen nu één voor één de belangrijke modelleringsconcepten van PSM de revue laten passeren, om zo een goede indruk te krijgen van de kracht van PSM. In de volgende paragraaf zullen we een aantal uitgebreidere voorbeelden zien.

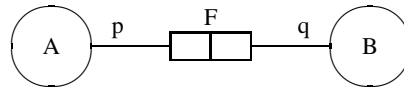


Figure 2: Een relatie getekend in de NIAM stijl

De tekenstijl die in dit verhaal voor PSM wordt gebruikt is een NIAM-achtige tekenstijl. Omdat het PSM een algemene basis is, zou hiervoor ook een ER- of INFOMOD-achtige tekenstijl gekozen kunnen worden. Ter illustratie staan in figuur 2 en 3 twee equivalente schema's in respectievelijk de NIAM- en de ER-tekenstijl. In een op PSM gebaseerd CASE tool mag men dan ook verwachten dat een datamodel via een aantal verschillende tekenstijlen weergegeven kan worden. (Zie voor een nadere uitwerking hiervan [HPW93].)

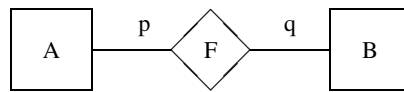


Figure 3: Een relatie getekend in de ER stijl

In figuur 2 maken we kennis met een aantal elementaire begrippen uit PSM. We zien daar de objecttypen  $A$  en  $B$ , de rollen  $p$  en  $q$ , die samen het feittype  $F$  vormen. Dit laatste is een fundamentele keuze in PSM: een feittype (ook wel aangeduid als relatie-type) is een verzameling van rollen, en legt, via deze rollen, een relatie tussen een aantal objecttypen. Met behulp van een *rolnaam* kan aangegeven worden in welke hoedanigheid de rol deze rol speelt.

Een directe consequentie is dat men bij elke rol kan aangeven tot welk feittype deze behoort. Verder hoort bij elke rol een uniek objecttype, namelijk dat objecttype dat door de rol in het feittype wordt opgenomen.

Een volgende keuze is dat feittypen als objecttypen beschouwd worden. Dit maakt het mogelijk om feittypen op te nemen in andere feittypen. Dit fenomeen wordt aangeduid als *objectificatie*.

Net als de meeste datamodellerings technieken, maakt men in PSM een strict onderscheid tussen abstracte en concrete objecttypen. Concrete objecttypen (ook wel aangeduid als labeltypen) kan men "opschrijven". Voorbeelden zijn: A-code (artikel-code) en het Sofi-nummer. Abstracte objecten daarentegen hebben deze eigenschap niet. Zo is een persoon een abstract object. Men kan een persoon als zodanig immers niet in een gegevensbank opnemen. Wat men wel kan, is personen als een abstract objecttype invoeren, en karakteriseren door een of meer attributen (bijvoorbeeld een persoonsnummer), waardoor men de abstracte objecten via een aantal concrete waarden kan *aanduiden*. De aanduidbaarheid van abstracte objecttypen is van cruciaal belang in elke modellerings techniek, dit is bekend onder de naam identificatie

Om het onderscheid tussen abstracte en concrete objecttypen aan te geven, worden in het PSM-diagram de namen van de concrete objecttypen tussen haakjes gezet. Deze, en andere tekenconventies vindt men in de appendix.

## Specialisatie

Specialisatie, ook bekend als subtypering, is een mechanisme waardoor men de mogelijkheid krijgt één of meer subtypen van een objecttype te definiëren. Voor de fijnproevers: deze subtypen hoeven niet disjunct te zijn. Karakteristiek voor specialisatie is dat het subtype een nadere verfijning is van het supertype. Dit betekent dat het subtype alle eigenschappen van het supertype *erft*. Daarnaast kan een subtype nog in extra relaties deelnemen. Deze laatste eigenschap is de eigenlijke reden waarom men subtypen zou willen invoeren. Een voorbeeld van een subtype-hierarchie is te zien in figuur 4. In dit figuur maken we ook

meteen kennis met grafische constraints. Het bolletje op het objecttype *Dier* staat voor een zogenaamde *totale-rol constraint*, en drukt uit dat elk dier van een bepaalde diersoort is. Het pijltje boven de rol *is-van* geeft een uniqueness constraint aan: een dier behoort tot *ten hoogste één* diersoort. Deze constraints zijn formeel ingevoerd in [BHW91], [HW92] en [WHB92].

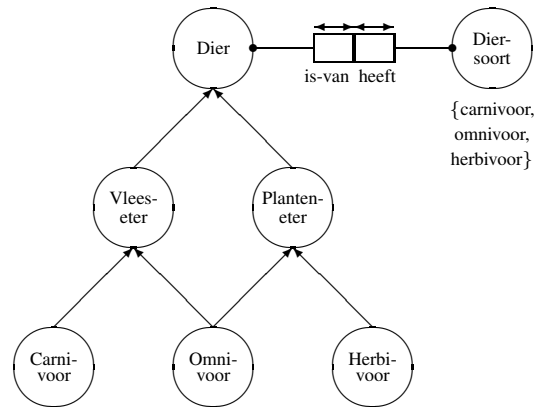


Figure 4: Een voorbeeld van een specialisatie hiërarchie

Om te kunnen bepalen welke instanties van het supertype terecht komen in het subtype is een zogenaamde *subtype-bepalende* regel nodig. In het voorbeeld van figuur 4 hebben deze regels de volgende gedaante:

*Vleeseter* = *Dier is-van Diersoort* {*carnivoor, omnivoor*}

*Planteneter* = *Dier is-van Diersoort* {*herbivoor, omnivoor*}

*Carnivoor* = *Dier is-van Diersoort* {*carnivoor*}

*Omnivoor* = *Dier is-van Diersoort* {*omnivoor*}

*Herbivoor* = *Dier is-van Diersoort* {*herbivoor*}

Deze subtype-bepalende regels zijn opgesteld in de taal LISA-D (Language for Information Structure and Access Descriptions), een semi-natuurlijke taal, met een formele grondslag, voor het specificeren en manipuleren van PSM datamodellen ([HPW93]).

## Generalisatie

In tegenstelling tot specialisatie, wordt bij generalisatie een objecttype gedefinieerd als de generalisatie van een stel onderliggende objecttypen (de *specifiers*). Een voorbeeld hiervan zien we in figuur 5. Dit is een schema voor een produktadministratie van *auto's* en *huizen*.

Generalisatie en specialisatie worden vaak met elkaar verward. Toch is er een fundamenteel onderscheid tussen deze beide begrippen. Bij generalisatie worden eigenschappen (zoals identificatie) opwaarts doorgegeven, dus van de specifiers naar het gegeneraliseerde objecttype. Bij specialisatie zagen we dat dergelijke eigenschappen neerwaarts worden doorgegeven. Typisch voor generalisatie is verder dat de specifiers onderling disjunct moeten zijn. Dus een instantie van een gegeneraliseerd object type komt maar uit één van zijn specifiers. Op die manier kan van elke object instantie uit het gegeneraliseerde type eenduidig de herkomst worden vastgesteld.

Als voorbeeld, kunnen we een eenvoudig toepassingsdomein bekijken, waarbij we te maken hebben met diverse soorten *Produkten*. *Produkten* kunnen *auto's* zijn of *huizen*. *Auto's* worden geïdentificeerd door een *registratienummer* en *huizen* door een combinatie van *huisnr* en *postcode*. Voor zowel *auto's* als *huizen* moet de aankoop prijs vastgelegd worden. Als we dit probleemgebied zouden modelleren door gebruik te maken van specialisatie (dit kan niet anders in bijvoorbeeld NIAM), dan krijgen we het schema

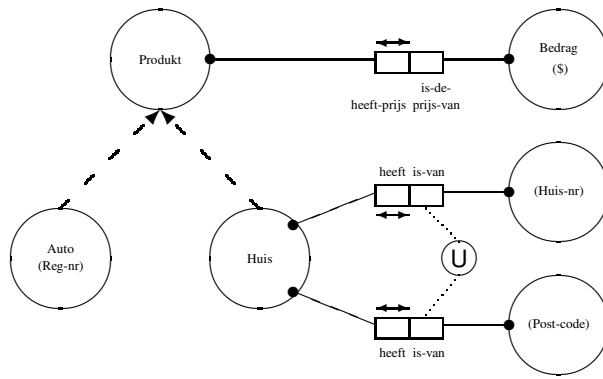


Figure 5: Produkt administratie met generalisatie

zoals weergegeven in figuur 6.

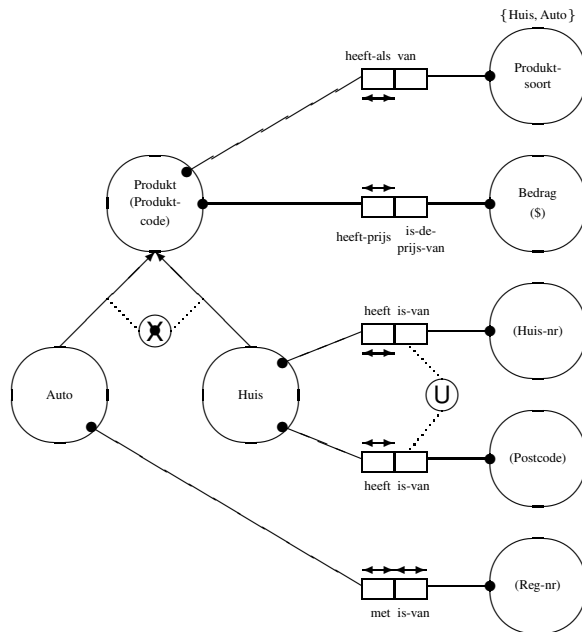


Figure 6: Overspecificatie: produkt administratie met specialisatie

Zoals we zullen aangeven, lijdt dit schema aan *overspecificatie*. Op de eerste plaats ontlene de subtypen *Auto* en *Huis* hun identificatie aan het supertype *Produkt*. Het is daarom nodig om een *Produktcode* in te voeren voor de identificatie van het objecttype *Produkt*. Van *Produktcode* was geen sprake in de oorspronkelijke domeinbeschrijving. De invoering hiervan in het applicatiedomein werd opgelegd door de modelleringstechniek. In sommige modelleringstechnieken is dit niet te vermijden, in PSM gelukkig wel.

Een tweede vorm van overspecificatie is het gevolg van de eis dat bij een subtypering een subtype-bepalende regel gegeven moet worden. Dit heeft in figuur 6 geleid tot een speciaal feittype en een speciaal objecttype *Produkttype*. De subtype-bepalende regels kunnen nu als volgt geformuleerd worden (met behulp van Lisa-D):

*Auto* = *Produkt is-van Produkt-soort* { *auto* }

*Huis* = *Produkt is-van Produkt-soort* { *huis* }

Ook nu werd de invoering van de extra objecttypen in het applicatiedomein afgedwongen door de

modelleringstechniek. Door de noodzaak om extra objecttypen in te voeren, die conceptueel niet relevant zijn, voldoet het schema uit figuur 6 niet aan de welbekende kwaliteitseis: het *conceptualisatieprincipe* (geen overspecificatie, zie b.v. [ISO87]).

Met behulp van generalisatie kunnen deze problemen van overspecificatie ondervangen worden. Zie daartoe weer het schema in figuur 5. Een *Produktcode* is nu niet nodig omdat het objecttype *Produkt* zijn identificatie erft van zijn specifiers *Auto* en *Huis*. Verder zijn er bij generalisatie geen subtype definierende regels nodig.

Het voorgaande voorbeeld suggereert dat specialisatie en generalisatie tegenhangers van elkaar zijn, en dat men steeds de keuze heeft welke van beide te kiezen. Dit echter is bepaald niet het geval. Zo zijn specialisatiestructuren met complexe subtype bepallende regels niet te modelleren met behulp van generalisatie. Aan de andere kant zijn complexe recursieve structuren niet te beschrijven met behulp van specialisatie.

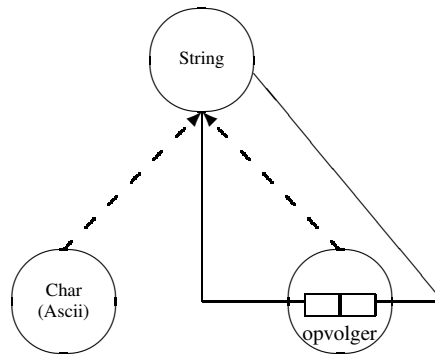


Figure 7: Een complexer voorbeeld van generalisatie

De combinatie van generalisatie en recursie, leidt tot verbluffende mogelijkheden. Beschouw als voorbeeld hiervan, figuur 7.

Een *string* bestaat uit een reeks van één of meer karakters. In dit voorbeeld wordt de identificatie van een instantie van het objecttype *string* dus verkregen uit de identificatie van de onderliggende instantie van *char* en de eventuele rest van de *string*. Dit voorbeeld illustreert overduidelijk dat met behulp van generalisatie, op éénvoudige wijze, recursieve objecttypen - objecttype welke gedefinieerd zijn in termen van zichzelf - gedefinieerd kunnen worden. Het voorbeeld uit figuur 7 is niet uit te drukken met behulp van een specialisatie.

## Powertypen

Een ander belangrijk modelleringsconcept in PSM is het *powertype*. Dit begrip is van belang in situaties waarin *groepen* van objecten zelf weer objecten zijn. De instanties van powertypen zijn groepen (verzamelingen) van instanties van een ander objecttype, het *elementtype* genaamd. Een powertype erft zijn identificatie van zijn elementtype, immers, om een groep van objecten te kunnen identificeren is het voldoende om de leden van die groep te kennen.

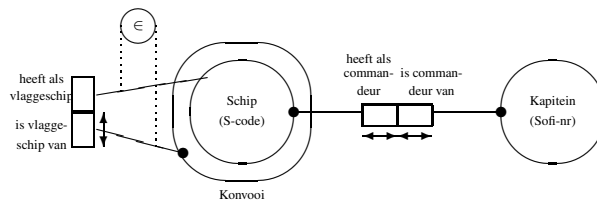


Figure 8: Het konvooi probleem

Een beroemd volgend voorbeeld van het gebruik van een powertype staat in figuur 8. In oorlogstijd varen vrachtschepen doorgaans in een konvooi. Zo'n konvooi bestaat uit een verzameling van schepen, wordt geleid door één schip. Elk schip wat in het konvooi vaart heeft een kapitein, die het schip commandeert. Ook voor dit voorbeeld geldt dat het schema niet ugedrukt kan worden met behulp van een traditionele modellering techniek zonder de introductie van surrogaat label typen ([HW93, HPW92]).

Als een ander, niet triviaal, voorbeeld van het gebruik van powertypen bekijken we een probleemgebied dat betrekking heeft op chemische reacties. Een chemische reactie heeft een aantal chemische stoffen in een, van te voren bepaalde, hoeveelheid nodig als invoer en produceert dan een aantal andere chemische stoffen, in vooraf berekende, hoeveelheden. Dit zou in ER gemodelleerd kunnen worden als in het schema in figuur 9. In dit figuur staat ook een voorbeeld instantie van dit schema. Om het schema niet al te complex te maken, worden de attributen van de verschillende entiteiten aangegeven door de attribuut naam (Natno), voorafgegaan door een hekje (#).

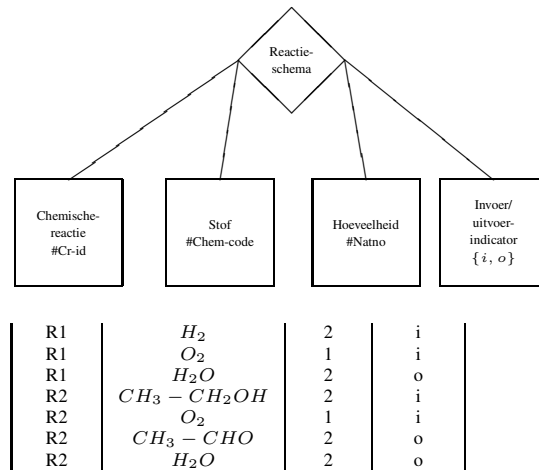


Figure 9: Chemische reacties in ER

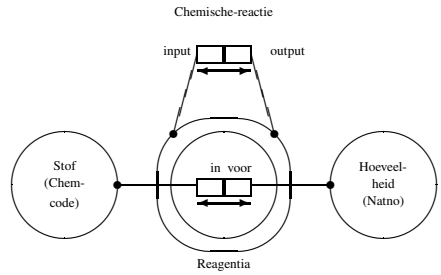
Het reactieschema van chemische reactie *R1* beschrijft de productie van waterdamp uit het mengsel van zuurstofgas en waterstofgas. Deze reactie gaat doorgaans met een explosie gepaard, vandaar ook dat het mengsel van zuurstofgas en waterstofgas ook bekend is als knalgas. Hierbij wordt meteen duidelijk dat het ER schema in figuur 9 mank gaat aan overspecificatie. Normaal zal men een reactie niet identificeren aan de hand van een code (*Cr-id*). Een reactie wordt in het algemeen geïdentificeerd door de invoer en uitvoerstoffen van de reactie, slechts in uitzonderlijke gevallen wordt een naam aan een reactie gegeven.

Naast de problemen rond de identificatie plicht, heeft het schema in figuur 9 nog een ernstig gebrek. Dit schema bevat een zogenaamde elementaire-updateanomalie. Als men een reactie invoert in het informatiesysteem, moet er voor elke invoer en uitvoer stof een tuple aan de instantiëring van de relatie worden toegevoegd. Dit betekent dat er tussentstanden ontstaan waarin de opgeslagen formule niet correct is! Dit natuurlijk moet vermeden worden. Met behulp van powertype, kan het domein van chemische reacties weergegeven worden met een schema zoals in figuur 10. Daar is voor het invoeren van een chemische reactie slechts één update nodig.

## Sequentietypen

Soms is niet alleen lidmaatschap van een groep van belang, maar ook de volgorde binnen de groep. Voor dit doel bevat PSM de zogenaamde *sequentietypen*. Bij veel hypermedia applicaties is er inderdaad behoefte aan zo'n mechanisme wat het maken van rijen ondersteund.

In figuur 7 hebben we al een schema gezien voor het modelleren van strings, dit schema kan met een sequentietype nog verder vereenvoudigt worden tot het schema in figuur 11. Ook in de komende voorbeelden zullen we zien dat de sequentietypen ons toestaan om elegante specificaties te maken van multimedia-applicaties.



<i>invoer</i>	<i>uitvoer</i>												
<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;"><i>voor</i></td> <td style="padding: 2px;"><i>in</i></td> </tr> <tr> <td style="padding: 2px;"><math>H_2</math></td> <td style="padding: 2px;">2</td> </tr> <tr> <td style="padding: 2px;"><math>O_2</math></td> <td style="padding: 2px;">1</td> </tr> </table>	<i>voor</i>	<i>in</i>	$H_2$	2	$O_2$	1	<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;"><i>voor</i></td> <td style="padding: 2px;"><i>in</i></td> </tr> <tr> <td style="padding: 2px;"><math>H_2O</math></td> <td style="padding: 2px;">2</td> </tr> </table>	<i>voor</i>	<i>in</i>	$H_2O$	2		
<i>voor</i>	<i>in</i>												
$H_2$	2												
$O_2$	1												
<i>voor</i>	<i>in</i>												
$H_2O$	2												
<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;"><i>voor</i></td> <td style="padding: 2px;"><i>in</i></td> </tr> <tr> <td style="padding: 2px;"><math>CH_3 - CH_2OH</math></td> <td style="padding: 2px;">2</td> </tr> <tr> <td style="padding: 2px;"><math>O_2</math></td> <td style="padding: 2px;">1</td> </tr> </table>	<i>voor</i>	<i>in</i>	$CH_3 - CH_2OH$	2	$O_2$	1	<table border="1" style="margin: auto;"> <tr> <td style="padding: 2px;"><i>voor</i></td> <td style="padding: 2px;"><i>in</i></td> </tr> <tr> <td style="padding: 2px;"><math>CH_3 - CHO</math></td> <td style="padding: 2px;">2</td> </tr> <tr> <td style="padding: 2px;"><math>H_2O</math></td> <td style="padding: 2px;">2</td> </tr> </table>	<i>voor</i>	<i>in</i>	$CH_3 - CHO$	2	$H_2O$	2
<i>voor</i>	<i>in</i>												
$CH_3 - CH_2OH$	2												
$O_2$	1												
<i>voor</i>	<i>in</i>												
$CH_3 - CHO$	2												
$H_2O$	2												

Figure 10: Chemische reactie in PSM

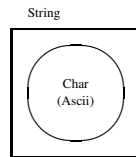


Figure 11: Een voorbeeld van een sequence type



## Schematypen

Een groot nadeel van bestaande datamodellerings technieken is het gebrek aan decompositie mogelijkheden. Het datamodel van veel applicaties beslaat al gauw een papier oppervlak waar men een aardig grote muur mee kan behangen. In PSM wordt decompositie mogelijk gemaakt door *schematypen*. Een schematype is een objecttype met een onderliggend schema. Dit schema moet een geldig PSM schema zijn.

Schematypen komen veelvuldig voor in *meta-modellen*. Een meta-model is een model van een modellerings techniek. Meta-modellen zijn vooral van belang voor het op een hoog niveau van abstractie definiëren van repository-structuren. Een repository (soms ook wel data dictionary of encyclopedia genoemd) vormt het hart van een CASE tool. In de repository van een CASE tool worden alle relevante modelleringsgegevens opgeslagen. Een goede structuur van de repository is zeer belangrijk omdat snel access op de gegevens daarbinnen noodzakelijk is. Dit geldt met name voor verificatie (voldoen de opgeslagen modellen wel aan de eisen van de betreffende modellerings techniek?).

Repository-structuren kunnen vaak zeer complex zijn. Een van de redenen hiervoor is dat decompositie veelvuldig voorkomt in modellerings technieken. Als een voorbeeld beschouwen we hier activiteitendiagrammen ([Sch84]), een eenvoudige variant van de veelvuldig gebruikte data flow diagrammen. In figuur 12 staan twee voorbeelden van activiteitendiagrammen. Activiteitendiagrammen bestaan uit activiteiten

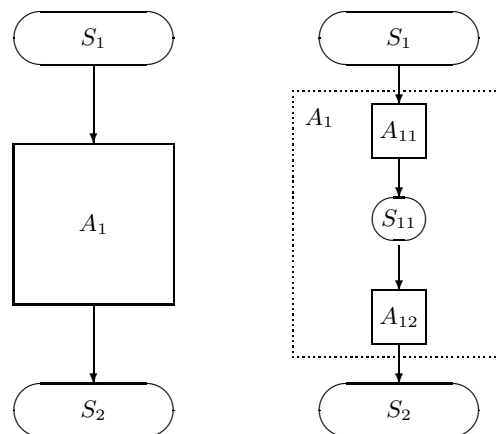


Figure 12: Een activiteitendiagram met decompositie

en toestanden. Activiteiten stellen processen voor, terwijl toestanden corresponderen met informatie. Toestanden kunnen zowel invoer als uitvoer van een activiteit zijn. Zowel activiteiten als toestanden kunnen gedeconponeerd worden in andere activiteitendiagrammen. In figuur 12 is het rechterschema de decompositie van activiteit  $A_1$  uit het linkerschema.

Het meta-model van activiteitendiagrammen is weergegeven als een PSM schema in figuur 13. In dit figuur is het objecttype *Activiteitendiagram* een schematype. Zoals blijkt uit het meta-model bestaat een activiteitendiagram uit een aantal activiteiten, toestanden en invoer en uitvoer relaties. Tevens is weergegeven dat zowel activiteiten als toestanden gedeconponeerd kunnen worden. Complexere voorbeelden van meta-modellen uitgedrukt in PSM, o.a. van data flow diagrammen, zijn te vinden in [HVNW92].

## 3 Modelleren van multimedia

Ook in het bedrijfsleven wordt er de laatste jaren, ten behoeve van de bouw van informatiesystemen, steeds vaker gebruik gemaakt van modellerings technieken zoals ER [Che76], INFOMOD [GJ84] en NIAM [NH89]. Het gebruik van dit soort technieken is dus geen pure academische exercitie meer. Ook de CASE tools ([McC89]) die deze technieken in mindere of meerdere mate ondersteunen worden steeds meer toegepast. Ten gevolge van deze ontwikkelingen, wordt het 'pure' Cobol programmeren vervangen

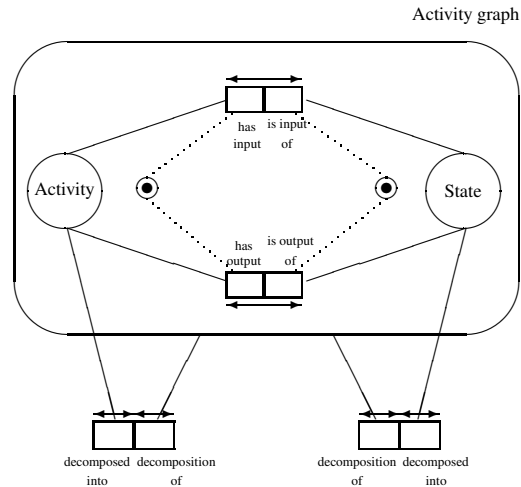


Figure 13: Meta-model van activiteitendiagrammen

door het werken in een 4e generatie taal. Soms biedt de gebruikte CASE tool zelfs de mogelijkheid de gemaakte datamodellen (automatisch) om te zetten naar “code”.

Deze traditionele technieken zijn evenwel niet in staat om de structuur van documenten te beschrijven zonder te zondigen tegen het conceptualisatie-principe. Beschouw ter illustratie van deze tekortkoming onderstaande beschrijving van hoe een boek in elkaar zit. We doen dit in de vorm van een syntax. Deze stijl is conform SGML (Standard General Markup Language), de de-facto industriële standaard voor dergelijke beschrijvingen ([ISO86]). Het symbool  $\rightarrow$  dient men uit te spreken als “bestaat uit”, en de komma als “gevolgd door”. De afkorting REP staat voor “een of meer herhalingen”.

boek  $\rightarrow$  titel, inhoud  
 inhoud  $\rightarrow$  hoofdstuk REP  
 hoofdstuk  $\rightarrow$  titel, paragrafen  
 paragrafen  $\rightarrow$  paragraaf REP  
 paragraaf  $\rightarrow$  string  
 string  $\rightarrow$  character REP

In figuur 14 staat deze syntax weergegeven als een ER-schema. De tekortkomingen hiervan zullen we verder niet bespreken, omdat het herhalingen zijn van de problemen die we eerder signalleerden.

In PSM komen we tot het schema in figuur 15. Het belang van deze informatiestructuur is dat we een directe koppeling hebben met de syntactische beschrijving van boeken. Omdat een elke syntactische beschrijving (conform de SGML-regels) kan worden omgezet naar een PSM-schema, is in PSM de zogenaamde *grammar-box* ingevoerd. Men kan nu volstaan met het geven van de syntax, het systeem draagt de zorg voor de omzetting naar een PSM-schema. In ons geval krijgen we figuur 16.

**Drs. A.H.M. ter Hofstede** is als onderzoeker verbonden aan het Software Engineering Research Centre (SERC) te Utrecht en aan de afdeling Informatiesystemen van de Katholieke Universiteit te Nijmegen.

**Drs. H.A. Proper** is als onderzoeker verbonden aan de afdeling Informatiesystemen van de Katholieke Universiteit te Nijmegen.

**Dr. ir. Th.P. van der Weide** is als universitair hoofddocent verbonden aan de afdeling Informatiesystemen van de Katholieke Universiteit te Nijmegen.

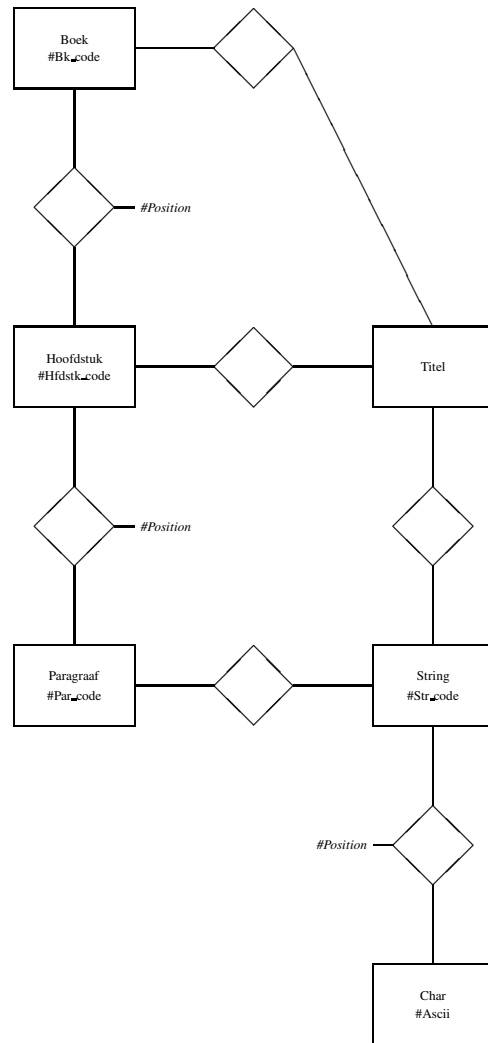


Figure 14: Een ER-schema voor de structuur van boeken

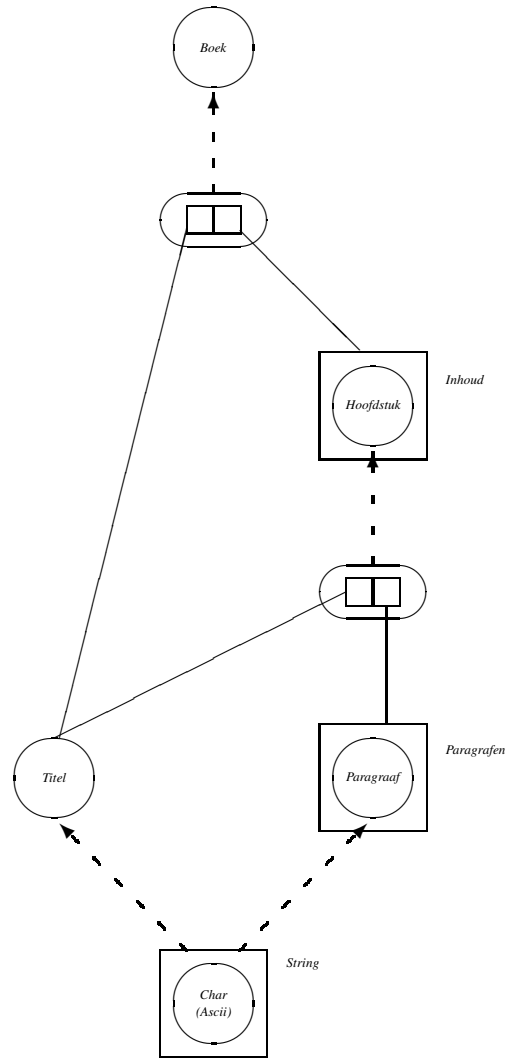


Figure 15: Een PSM-schema voor de structuur van boeken

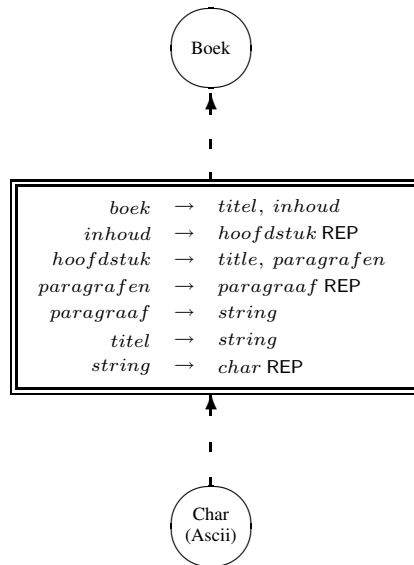






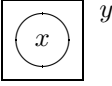

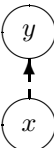
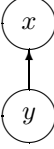
Figure 16: Het gebruik van de grammar box

## A Legenda van graphische symbolen

Deze appendix bevat een overzicht van de in dit artikel gebruikte symbolen voor het modelleren van objecttypen, de conventies voor generalisatie en specialisatie en de graphische representatie van constraints. Voor een meer uitgebreide behandeling van de achterliggende concepten, zie: [HW93, HPW92].

## References

- [BHW91] P. van Bommel, A.H.M. ter Hofstede, and Th.P. van der Weide. Semantics and verification of object-role models. *Information Systems*, 16(5):471–495, October 1991.
- [Che76] P.P. Chen. The entity-relationship model: Towards a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, March 1976.
- [GJ84] J.J. van Griethuysen and D.A. Jardine. De infomod-benadering van informatiemodellering. *Informatie*, 26(6):409–500, 1984. In Dutch.
- [HPW92] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Data Modelling in Complex Application Domains. In P. Loucopoulos, editor, *Proceedings of the Fourth International Conference CAiSE'92 on Advanced Information Systems Engineering*, volume 593 of *Lecture Notes in Computer Science*, pages 364–377, Manchester, United Kingdom, EU, May 1992. Springer Verlag, Berlin, Germany, EU. ISBN 3540554815
- [HPW93] A.H.M. ter Hofstede, H.A. Proper, and Th.P. van der Weide. Formal definition of a conceptual language for the description and manipulation of information models. *Information Systems*, 18(7):489–523, October 1993.
- [HVNW92] A.H.M. ter Hofstede, T.F. Verhoef, E.R. Nieuwland, and G.M. Wijers. Specification of Graphic Conventions in Methods. In B. Theodoulidis and A. Sutcliffe, editors, *Proceedings of the Third Workshop on the Next Generation of CASE Tools*, pages 185–215, Manchester, United Kingdom, May 1992.
- [HW92] A.H.M. ter Hofstede and Th.P. van der Weide. Formalisation of techniques: chopping down the methodology jungle. *Information and Software Technology*, 34(1):57–65, January 1992.

	<i>objecttype</i>
	<i>labeltype x</i>
	<i>rol</i>
	<i>y is een powertype van x</i>
	<i>y is een sequentietype van x</i>
	<i>grammar box</i>
	<i>y is een generalisatie van x</i>
	<i>y is een specialisatie van x</i>

$\longleftrightarrow$	<i>uniciteitsconstraint over een enkelvoudig feittype</i>
$\textcircled{u}$	<i>uniciteitsconstraint over meerdere feittypen</i>
$\bullet$	<i>total role of cover constraint</i>
$\textcircled{n..m}$	<i>cardinaliteitsconstraint</i>
$\otimes$	<i>exclusiviteitsconstraint</i>
$\textcircled{\in}$	<i>lidmaatschapsconstraint</i>
$\textcircled{\subseteq}$	<i>deelverzamelingsconstraint</i>
$\textcircled{=}$	<i>gelijkheidsconstraint</i>
$\textcircled{\{x_1..x_k\}}$	<i>opsommingsconstraint</i>

- [HW93] A.H.M. ter Hofstede and Th.P. van der Weide. Expressiveness in conceptual data modelling. *Data & Knowledge Engineering*, 10(1):65–100, February 1993.
- [ISO86] *Information Processing – Text and Office Systems – Standard General Markup Language (SGML)*, 1986. ISO 8879:1986.  
<http://www.iso.org>
- [ISO87] *Information processing systems – Concepts and Terminology for the Conceptual Schema and the Information Base*, 1987. ISO/TR 9007:1987.  
<http://www.iso.org>
- [McC89] C.L. McClure. *CASE is Software Automation*. Prentice-Hall, Englewood Cliffs, New Jersey, 1989. ISBN 0131193309
- [NH89] G.M. Nijssen and T.A. Halpin. *Conceptual Schema and Relational Database Design: a fact oriented approach*. Prentice-Hall, Sydney, Australia, 1989. ASIN 0131672630
- [Sch84] G. Scheschonk. *Eine auf Petri-Netzen basier-en-de Konstruk-tion-s, Ana-ly-se und (Teil)-Veri-fica-tion-s-me-tho-de zur Modellierungsunterstützung bei der Entwicklung von Informationssystemen*. PhD thesis, Berlin University of Technology, Berlin, Germany, 1984. (In German).
- [Sch89] H. Schouten. SGML\*CASE89–The storage of documents in Data Bases. Technical Report 03-11, TFDL/ECIT, PO 356, 6700 Wageningen, The Netherlands, 1989.
- [WHB92] Th.P. van der Weide, A.H.M. ter Hofstede, and P. van Bommel. Uniquet: Determining the Semantics of Complex Uniqueness Constraints. *The Computer Journal*, 35(2):148–156, April 1992.