

Model-based Application Development

Model Driven Development

Modelleren en genereren van informatiesystemen

Kees Kranenburg, Ad van Riel

Syllabus voor studenten

Uitgave 2007

Inhoud

Ten geleide	4
1 MAD: introductie	6
1.1 Model-based Application Development.....	6
1.2 De vertrouwde Software Engineering-benadering	6
1.3 Applicatiegeneratie.....	9
1.4 De MAD-benadering	11
1.5 Model Driven Architecture.....	14
1.6 Samenvatting	18
2 De MAD-modellencyclus.....	19
2.1 De modellencyclus	19
2.2 Het bedrijfsactiviteitenmodel	20
2.3 Het informatiemodel.....	22
2.4 Systeemafbakening.....	26
2.5 Het gegevensmodel	28
2.6 Functionaliteit.....	33
2.7 Samenvatting	41
3 De MAD-projectaanpak	42
3.1 Voorafgaand aan een MAD-project	42
3.2 Projectplanning en -beheersing	44
3.3 Projectinrichting	48
3.4 Projectuitvoering	50
3.5 Onderhoud	56
3.6 Samenvatting	57
4 De template-benadering.....	60
4.1 Definitie.....	60
4.2 Hergebruik van specificaties	61
4.3 Moderne ontwikkelomgeving.....	61
4.4 Acquisitie van de template	63
4.5 Aanpassen en invoeren van de template.....	63
4.6 Samenvatting	65
5 De Actimod-methode	66
5.1 Algemene concepten	66
5.2 Decompositie.....	68
5.3 Conventies	71

6	De Entity-Relationship-methode	76
6.1	Algemene concepten	76
6.2	De ISO-variant.....	79
6.3	Ter aanvulling.....	82
7	Naar een relationeel gegevensmodel	86
7.1	Terminologie van het relationele model.....	86
7.2	Naar een relationeel gegevensmodel door normaliseren.....	88
7.3	Naar een relationeel gegevensmodel door informatiemodellering.....	92
	Appendix A – Casus: de boekenclub.....	97
A.1	Het bedrijfsactiviteitenmodel	97
A.2	Het informatiemodel.....	100
A.3	Het specificatiemodel	103
	Appendix B – Casus: de fabriek.....	106
B.1	Het bedrijfsactiviteitenmodel	106
B.2	Het informatiemodel.....	108
B.3	Het specificatiemodel	112
B.4	Een regel nader uitgewerkt.....	114
	Literatuur	118

Ten geleide

Doel en doelgroep

Het boek¹ heeft, zoals de titel al aangeeft, ten doel de lezer inzicht te geven in het modelmatig ontwikkelen van applicaties. Het accent van dit boek ligt op het modelleren, de onderlinge samenhang van modellen en het hergebruik van modellen.

Dit boek richt zich in het bijzonder op personen betrokken bij de ontwikkeling van informatiesystemen. Hierbij wordt gedacht aan personen die vanuit hun functie – projectleider, technisch architect, informatieanalist, systeemontwikkelaar – professioneel bezig zijn met software-engineering. Maar ook voor personen uit de gebruikersorganisatie – kader en gebruikersvertegenwoordigers die inhoud moeten geven aan functionele eisen van applicaties – is dit boek van belang.

Bovendien richt het zich op de cursisten van de module ‘Informatie-analyse en -modellering (HSB.1)’, dat deel uitmaakt van het Ambi-diploma en van de Academy Track Software Design and Development van I-Tracks.

Het boek is tevens geschikt als studiemateriaal voor informaticastudenten die zich willen verdiepen in moderne aanpakken voor systeemontwikkeling met behulp van geavanceerde, op Model Driven Architecture gebaseerde ontwikkelomgevingen.

Toepassingsgebied

Model-based Application Development (MAD) is in eerste instantie een ontwikkelaanpak voor het zelf ontwikkelen van applicaties (maatwerk), maar moet ook als een serieus alternatief worden beschouwd in die situaties waar de inzet van standaard applicatiepakketten voor de hand ligt. Dit laatste is zeker het geval wanneer er herbruikbare modellen voor het betreffende applicatiegebied beschikbaar zijn.

MAD kan worden toegepast voor de ontwikkeling van administratieve informatiesystemen en technische informatiesystemen en kan als hulp dienen bij de specificatie van embedded software. Het boek is zowel geschikt voor organisaties in de rol van ‘opdrachtgever’ als voor organisaties in de rol van ‘opdrachtnemer’. MAD is geschikt als aanpak bij een gegevensgeoriënteerde benadering en bij een objectgeoriënteerde benadering voor softwareontwikkeling.

¹ Deze syllabus is voor studenten samengesteld op basis van de inhoud van het originele boek (2^e druk, 2004). De syllabus is gratis verkrijgbaar via de website van de Stichting EXIN (www.exin.nl).

Terminologie

De concepten van MAD gelden voor een gegevensgeoriënteerde benadering en voor een objectgeoriënteerde benadering. In dit boek worden de termen uit beide benaderingen gehanteerd volgens onderstaand schema (gebaseerd op de 3-schema-architectuur).

	Gegevensgeoriënteerde benadering	Objectgeoriënteerde benadering
Conceptueel	Informatiemodel	Business domeinmodel
Extern schema	Gegevensmodel	Objectmodel
Intern schema	Databaseschema	Classes/Databaseschema

Over de auteurs

Kees Kranenburg is als portfolio & competence manager werkzaam bij het Software Development & Maintenance Center, de moderne softwarefabriek van Atos Origin (www.atosorigin.nl). Hij geeft richting en leiding aan de innovatie van de diensten en competenties op gebied van softwareontwikkeling, applicatie-integratie, migraties, testen en applicatiebeheer. Email: kees.kranenburg@atosorigin.com

Ad van Riel is in dienst van Philips Corporate Information Technology. Hij houdt zich bezig met innovatie van informatietechnologie en de positionering daarvan in het concernbeleid.

1 MAD: introductie

In dit hoofdstuk belichten we de kenmerken van modelmatige systeemontwikkeling.

1.1 Model-based Application Development

Model-based Application Development (MAD) is een aanpak voor systeemontwikkeling en wordt gekenmerkt door:

- een modelmatige Software Engineering-benadering;
- specificaties van waaruit systemen worden gegenereerd;
- een incrementele en iteratieve ontwikkelaanpak;
- intensieve gebruikersparticipatie;
- hergebruik van modellen.

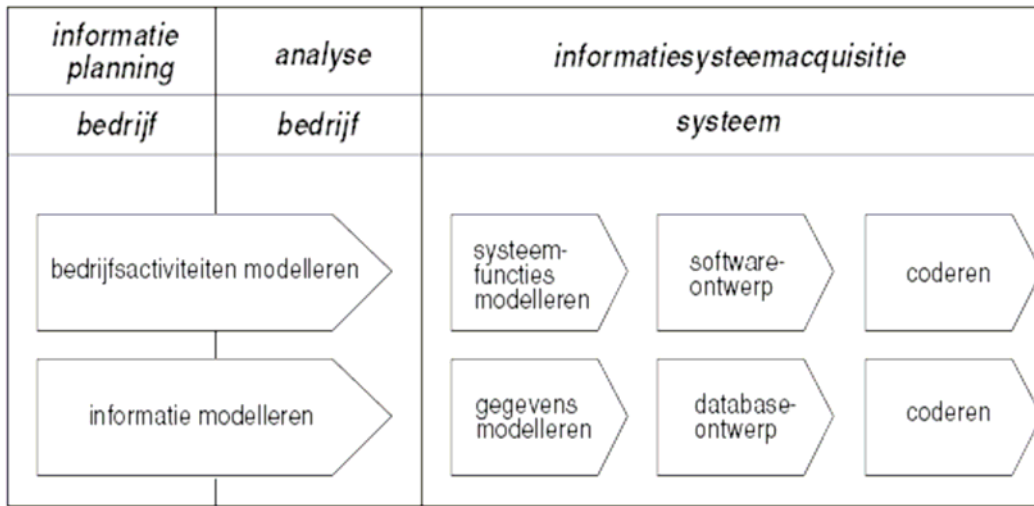
MAD is een inhoudelijke invulling van klassieke en moderne aanpakken voor softwareontwikkeling, waarbij de nadruk sterk op modelleren ligt, en waarbij applicaties vanuit een modelmatig gepopuleerde repository worden gegenereerd.

MAD kan worden toegepast in combinatie met een iteratieve aanpak, bijvoorbeeld Dynamic Systems Development Method of Unified Process.

1.2 De vertrouwde Software Engineering-benadering

Modelleren speelt een belangrijke rol bij systeemontwikkeling. Wij zien modelleren als een proces waarbij verschillende transformaties, zoals ordening, vertaling en verfijning, worden uitgevoerd op een initiële specificatie om ten slotte werkende machinecode en een correcte interactie met de gebruikers- en systeemomgeving op te leveren. Figuur 1 laat de verschillende modelleringactiviteiten zien die hiervoor nodig zijn:

- modelleren van bedrijfsactiviteiten;
- modelleren van informatie;
- systeemafbakening;
- modelleren van systeemfuncties;
- modelleren van gegevens;
- softwareontwerp;
- databaseontwerp;
- codering.



Figuur 1 Modelleringsactiviteiten

Modelleren van bedrijfsactiviteiten

Het doel van het modelleren van bedrijfsactiviteiten is inzicht krijgen in de bedrijfsactiviteiten en hun informatie-uitwisseling. Het is een bedrijfsgerichte analyse-activiteit. De resultaten worden vastgelegd in een bedrijfsactiviteitenmodel. Dit model wordt tevens gebruikt om de bedrijfsactiviteiten te kunnen aanwijzen die voor automatisering in aanmerking komen. Daarnaast is het modelleren van bedrijfsactiviteiten nuttig in de communicatie met gebruikers.

Modelleren van informatie

In het informatiemodel wordt de informatiebehoefte van de bedrijfsactiviteiten vastgelegd. Dit model beschrijft de entiteitstypen, attribuuttypen en informatieregels. Het doel van informatiemodellering is het structureren van de informatiebehoefte. Het is een bedrijfsgerichte analyseactiviteit.

Systeemafbakening

De omvang van het beoogde geautomatiseerde informatiesysteem wordt vastgelegd met behulp van een systeem-contextmodel. Dit model toont het te ontwikkelen geautomatiseerde informatiesysteem als een 'black box' met interfaces met andere informatiesystemen. Het is een systeemgerichte specificatie-activiteit.

Modelleren van systeemfuncties

Door het modelleren van systeemfuncties wordt de functionaliteit van het geautomatiseerde informatiesysteem gedefinieerd in een procesmodel. Dit gebeurt meestal in een aantal verschillende niveaus van abstractie. Dit model toont, in de vorm van een netwerkmodel:

- de processen;
- het gedrag van het geautomatiseerde informatiesysteem;
- de daarbij benodigde gegevens.

Het is een systeemgerichte specificatie-activiteit.

Modelleren van gegevens

De in het informatiemodel vastgelegde informatiebehoefte wordt vertaald naar een structuur en representatie van gegevenstypen. Gegevens modelleren is een systeemgerichte specificatieactiviteit.

Softwareontwerp

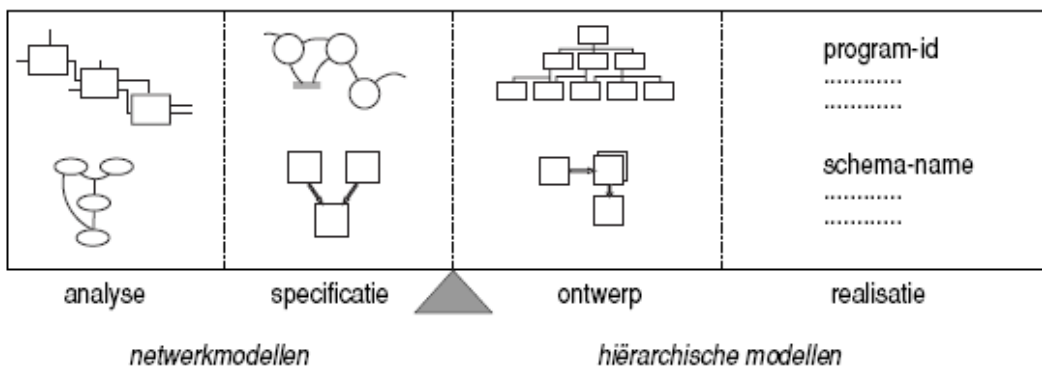
Software-structuurmodellen tonen de hiërarchie van de softwaremodulen en de gegevens die onderling worden gecommuniceerd. Programmaspecificaties completeren het ontwerp.

Databaseontwerp

Het gegevensmodel wordt getransformeerd naar structuren voor het opslaan en opvragen van gegevens, zoals die door het te gebruiken databasemanagementsysteem (DBMS) worden ondersteund.

Coderen

De software- en databasespecificaties worden vertaald in een programmeertaal en een data-definitietaal.



Figuur 2 Modellencyclus

Figuur 2 geeft schematisch de hierboven beschreven modellencyclus weer, zoals deze is ontstaan vanaf eind jaren '70. De kenmerken van deze modellencyclus zijn:

- Er is een scheiding tussen gegevens- en procesmodellen.
- De gelijkvormigheid van de modellen verdwijnt in de cyclus. Er is een omslagpunt van netwerkmodellen naar hiërarchische modellen (in figuur 2 aangegeven met een driehoek). Hier speelt de creativiteit van de ontwerper een doorslaggevende rol, wat in het algemeen later onderhoud aan het systeem moeilijk maakt. Immers het systeemmodel lijkt meestal maar nauwelijks op het (informatie)model van de werkelijkheid.
- Er is sprake van faseovergangen met 'complete' modellen (waterval-projectscenario). De ervaring heeft ons geleerd dat het zeer moeilijk, zo niet onmogelijk is, om samen met gebruikers complete specificaties te maken in een aparte, afgeronde fase. Meestal is het zo dat, als het systeem er eenmaal is, er alsnog een stortvloed van aanvullende en gewijzigde specificaties los komt. Dit komt dan vaak ten laste van het

onderhoudsbudget, en niet van het projectbudget, wat een vertekend beeld van de werkelijkheid geeft.

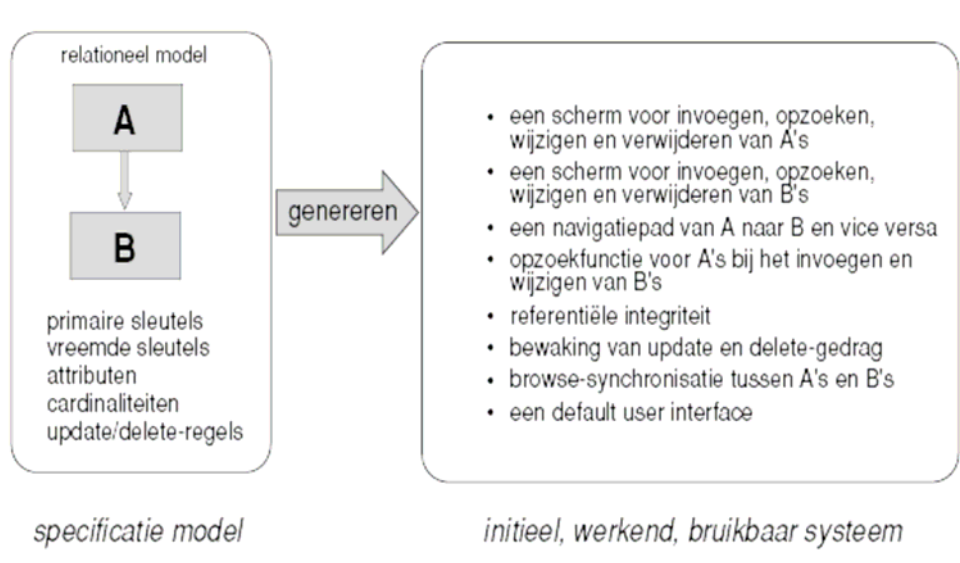
1.3 Applicatiegeneratie

In de beginjaren '90 verschijnen er gereedschappen op de markt die delen van het ontwikkeltraject automatiseren: generatoren die uit specificaties een werkende applicatie genereren. Het verschijnsel 'applicatiegeneratie' is gebaseerd op de volgende overwegingen:

- Informatiesystemen hebben veel functionaliteit gemeen, zoals onderhoudsfuncties voor iedere tabel in de database, navigatiepaden door de database en menubesturing in een aantal varianten. Deze functionaliteit kan als 'triviaal' worden beschouwd en kan worden gegenereerd.
- Het door het systeem afgedekte universum van discussie maakt het systeem essentieel verschillend van andere systemen. (Het 'universum van discussie' is de verzameling van alle dingen, situaties en gebeurtenissen waarover geïnformeerd wordt. Het universum van discussie is een gedeelte van een reële of hypothetische wereld [Griethuysen].
- Het systeem kan uiteindelijk worden gemodelleerd in een relationeel gegevensmodel of een objectmodel, aangevuld met expliciete functionele specificaties in de vorm van gedragsregels.

Op basis van een relationeel gegevensmodel of een objectmodel kan een generator een werkend systeem genereren met de volgende (triviale) functionaliteit (figuur 3):

- de databasestructuur;
- een keuzemenu;
- een scherm voor elke tabel uit het gegevensmodel voor het invoegen, wijzigen en verwijderen van gegevens;
- navigatiepaden tussen alle tabellen volgens de structuur van het gegevensmodel;
- zoekfunctionaliteit voor elke 'owner'-tabel;
- browse-synchronisatie op een 'owner/member'-combinatie;
- bewaking van de referentiële integriteit;
- bewaking van de 'update & delete'-regels.



Figuur 3 Gegeneerde 'triviale' functionaliteit

Als een generator de bovengenoemde functionaliteiten kan genereren, is er dus een werkend en ook een bruikbaar systeem. De gehele database is benaderbaar en bruikbaar. Een dergelijk systeem, gebaseerd op een goed datamodel, kan dienen als prototype met behulp waarvan de overige functionaliteit kan worden gespecificeerd.

Applicatiegeneratoren worden – idealiter – gekenmerkt door:

- een zuivere 3-schema-architectuur;
- een actieve of dynamische repository (bij een actieve repository worden de regels uitgeneraald in de broncode; bij een dynamische repository worden de regels op runtime geïnterpreteerd);
- de mogelijkheid om logische gegevensmodellen of objectmodellen af te beelden;
- voorgedefinieerde events op conceptueel en extern schaniveau;
- de mogelijkheid om business rules te definiëren;
- de mogelijkheid om een systeem te genereren naar diverse platforms.

De 4GL-applicatiegeneratoren uit de beginjaren '90 zijn opgevolgd door een nieuwe generatie ontwikkelomgevingen, die aan de hand van objectgeoriënteerde modellen code genereren voor het J2EE- en Microsoft .NET-platform.

Bij de toepassing van applicatiegeneratoren verschuift de focus van softwareontwikkeling naar de analyse en specificatie: het modelleren van de bedrijfswerkelijkheid en de gewenste systeemfunctionaliteit. Hierdoor zal het verschil tussen een gegevensgeoriënteerde benadering en een objectgeoriënteerde benadering afnemen, omdat de verschillen tussen beide benaderingen zich vooral uiten in het ontwerp en de realisatie van software. In de analyse en specificatie vertonen beide benaderingen meer overeenkomsten dan verschillen en verschilt een objectgeoriënteerde analyse niet wezenlijk van de 'klassieke' informatiemodellering (entiteitstypen, relaties en informatieregels).

Ofschoon er technische verschillen bestaan tussen deze ontwikkelomgevingen, is er een trend waarneembaar dat dit soort tools zich in een zelfde richting ontwikkelt, vooral voor wat de aard en omvang van de specificaties betreft waarmee ze worden gevoed. Een belangrijke aanzet hiertoe is Model Driven Architecture (MDA). Paragraaf 1.5 gaat hier nader op in.

1.4 De MAD-benadering

Modelleren

De inzet van applicatiegeneratoren heeft invloed op de modellencyclus zoals deze hiervoor is besproken (zie figuur 1). Vanaf het moment van opstellen van het gegevensmodel wordt het effect hiervan al merkbaar. Zoals gezegd wordt een generator gevoed met een gegevensmodel/objectmodel met aanvullende expliciete specificaties, op basis waarvan een werkend systeem wordt gegenereerd. Daarmee vervalt uit figuur 1:

- softwareontwerp;
- databaseontwerp;
- codering.

Bij de Software Engineering-benadering in een MAD-aanpak ligt de nadruk met name op het modelleren van bedrijfsactiviteiten, informatie en gegevens (figuur 4). In hoofdstuk 2 worden deze modeltypen en hun onderlinge samenhang nader toegelicht.

In de praktijk krijgen ook nu nog bedrijfs- en informatiemodellen niet altijd de aandacht die ze verdienen. Vaak wordt al begonnen met ontwerp en realisatie terwijl de analyse- en specificatiemodellen nog niet accuraat en volledig zijn. Fouten en omissies worden dan later in de software gerepareerd. Het gebruik van generatoren brengt hierin verandering. Omdat generatoren worden gevoed met specificaties in de vorm van modellen, wordt modelleren tijdens analyse en specificatie opeens voor de hand liggend of zelfs afgedwongen.

Het modelleren van systeemfunctionaliteit wordt volledig anders. Er wordt onderscheid gemaakt tussen triviale functionaliteit en expliciete functionaliteit. De triviale functionaliteit, zoals onderhouds- en navigatiefuncties, wordt gegenereerd op basis van het gegevensmodel/objectmodel. De expliciete functionaliteit wordt op een bepaalde manier gespecificeerd. In hoofdstuk 2 wordt dit nader uitgewerkt.

Incrementeel en iteratief ontwikkelen

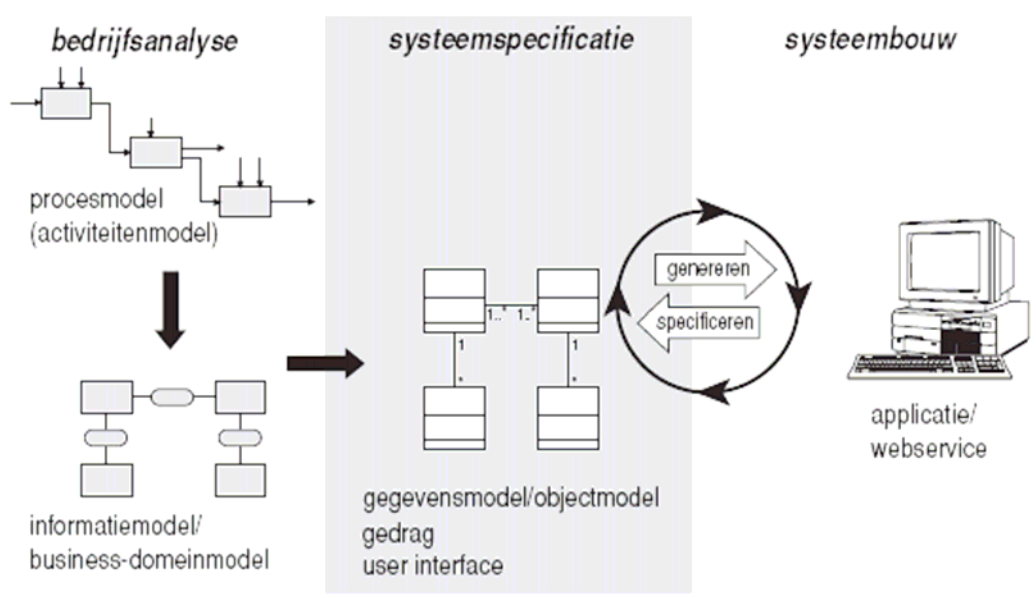
In MAD wordt incrementeel en iteratief ontwikkeld.

Bij een incrementele ontwikkelbenadering wordt het systeem opgedeeld in kleine, overzichtelijke systeemdelen, die vervolgens parallel of na elkaar worden ontwikkeld. Een increment is een logische eenheid van functionaliteit die kan worden geïsoleerd, ontwikkeld en geïmplementeerd.

In een MAD-aanpak wordt de bedrijfsanalyse, resulterend in activiteitenmodellen en informatie(sub)modellen, uitgevoerd over de volle breedte van het applicatiegebied. Incrementen worden herkend op basis van het informatie(sub)model. De volgorde waarin de incrementen verder worden gedetailleerd en ontwikkeld wordt bepaald op basis van:

- afhankelijkheden tussen de incrementen;
- prioriteiten van de gebruikers.

Bij een iteratieve ontwikkelbenadering worden de ontwikkelfasen in een aantal iteraties snel achter elkaar doorlopen.



Figuur 4 Model-based Application Development (MAD)

Figuur 4 geeft schematisch de iteratieve aanpak bij MAD weer. Uitgaande van een procesmodel wordt in samenspraak met materiedeskundigen een informatiemodel of een business-domeinmodel opgesteld. Dit model is nog globaal van aard: het toont de dingen uit de werkelijkheid (entiteitstypen of business-objecttypen), hun kenmerken identificerende attribuuttypen en enkele kenmerkende attribuuttypen) en het op dat moment bekende en voor de hand liggende gedrag (informatieregels, business rules). Van dit informatiemodel of business-domeinmodel wordt een (relationeel) gegevensmodel of een meer gedetailleerd objectmodel (klassendiagram) afgeleid. Dit model wordt aan de applicatiegenerator aangeboden, die er een eerste systeemversie uit genereert.

Hierna volgt een iteratief ontwikkelproces van prototyping en verdere detaillering van specificaties: door het tonen en bespreken van de gegenereerde systeemversie worden de al bekende informatieregels gevalideerd en de nog ontbrekende functionaliteit gespecificeerd. Deze specificaties worden weer toegevoegd aan de generator, die er een volgende systeemversie uit genereert. Eventuele tekortkomingen van het informatiemodel of business-domeinmodel, die in deze fase aan het licht komen, worden hersteld.

De MAD-projectaanpak wordt in hoofdstuk 3 nader uitgewerkt.

Joint Application Design

In MAD wordt bij de gebruikersparticipatie gebruikgemaakt van een techniek die Joint Application Design wordt genoemd.

Joint Application Design (JAD) is een techniek gebaseerd op intensieve en interactieve workshops waar zowel automatiseerders als gebruikersvertegenwoordigers in participeren. JAD stamt uit de eindjaren '70 (IBM). JAD is bedoeld om een goede, effectieve communicatie te bewerkstelligen tussen de informatiespecialist en de gebruikers.

JAD kan voor verschillende doeleinden worden toegepast. Op het niveau van informatieplanning kunnen JAD-sessies gebruikt worden om de huidige bedrijfsprocessen in kaart te brengen. Op systeemniveau kunnen JAD-sessies bijdragen om specificaties voor een specifieke applicatie te definiëren.

Een JAD-sessie wordt geleid door een sessieleider, die verantwoordelijk is voor de bepaling van de doelstellingen van iedere sessie en voor de realisatie van die doelstellingen. De sponsor (opdrachtgever) definieert de bedrijfsdoelstellingen en initieert de sessies. Een of meer gebruikersvertegenwoordigers verwoorden de eisen en wensen, die door de informatiespecialisten worden vertaald naar bedrijfs- en systeemmodellen. Een secretaris doet verslag van de sessies en verzorgt de documentatie. Specialisten, zowel van de gebruikerskant als vanuit de automatisering, kunnen tijdens de sessies worden geraadpleegd.

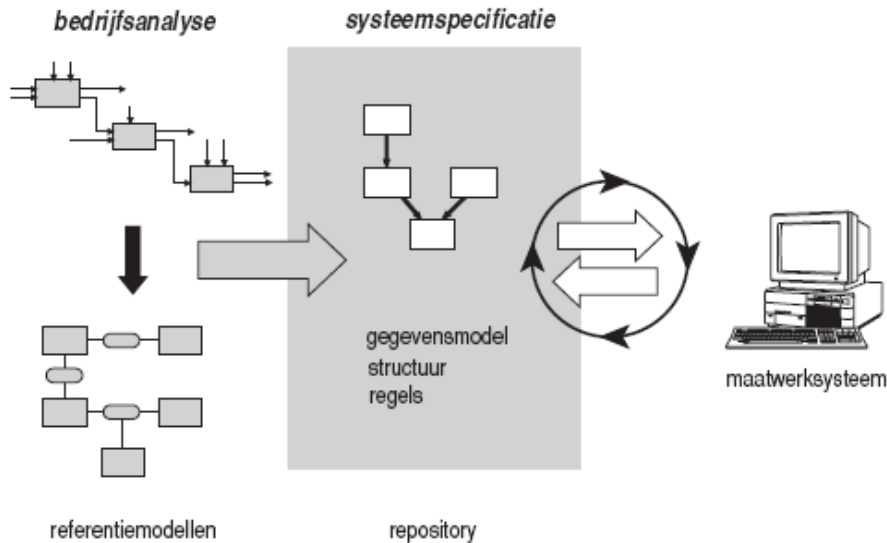
De voorbereiding van de sessie, de sessie zelf en de follow-up vinden plaats volgens een strikte agenda en procedures.

JAD beoogt een versnelling van het analyse- en specificatieproces door consolidatie van probleemdefinitie, bedrijfsanalyse en specificaties en de inzet van de juiste personen. JAD-sessies vervangen het 'eindeloos' heen-en-weer-afstemmen van specificaties in bilaterale gesprekken. Hierdoor wordt een verkorting van de doorlooptijd voor het analyse- en specificatietraject bewerkstelligd. De intensieve gebruikersbetrokkenheid komt de kwaliteit van specificaties ten goede en leidt tot een betere acceptatie van het systeem.

Joint Application Design en hieraan gerelateerde technieken, onder andere time boxing en prioriteiten toekennen aan systeemfunctionaliteit, zijn opgenomen in Dynamic Systems Development Method (DSDM Consortium).

Applicatie-templates

In toenemende mate worden er standaard applicatiepakketten in de vorm van CASE-modellen aangeboden. Daarbij worden benamingen gebruikt als rompsystemen, designware, kernel systems en casco-systemen. Wij sluiten ons aan bij meest gebruikte term: applicatie-templates.



Figuur 5 De template-benadering

Applicatie-templates zijn specificatiemodellen opgeslagen in de repository van een applicatiegenerator. Hiervan uitgaand wordt een werkend informatiesysteem gegenereerd. Referentiemodellen kunnen aan de template ten grondslag liggen (figuur 5). Het werkende informatiesysteem kan ook als prototype fungeren en is op eenvoudige wijze aan te passen aan lokale, specifieke eisen en wensen.

Aanpassing vindt plaats in een moderne ontwikkelomgeving. CASE-tools en applicatiegeneratoren ondersteunen een standaardwijze van werken en bewerkstelligen een hoge kwaliteit en een hoge productiviteit. De applicatie kan worden gegenereerd voor verschillende hardware- en softwareplatforms.

Bij deze aanpak – hergebruik van modellen in de vorm van referentiemodellen en een daarvan afgeleide gevulde repository – wordt de flexibiliteit van zelfbouw gecombineerd met de snelle beschikbaarheid van applicatiepakketten. Daarnaast geeft het een organisatie een vliegende start en verhoogt het de efficiëntie van analyse en specificatie. Deze wijze van systeemontwikkeling vormt een aantrekkelijk alternatief voor de inzet van standaard applicatiepakketten.

In hoofdstuk 4 gaan we nader in op de template-benadering.

1.5 Model Driven Architecture

Het modelleren en genereren van applicaties, zoals dit met behulp van de 4GLapplicatiegeneratoren wordt toegepast, heeft een nieuwe impuls gekregen door Model Driven Architecture (MDA). MDA is ontwikkeld door de Object Management Group en definieert een raamwerk van modellen voor softwareontwikkeling. Om de complexiteit van applicaties te kunnen beheersen voorziet MDA in een strikte scheiding van de logica van een systeem in een platformonafhankelijk model en de implementatie ervan in een

platformspecifiek model. Dankzij deze opzet kan de ‘business intelligence’ van een organisatie worden opgeslagen (en onderhouden) in een gestandaardiseerd, eenduidig en goed gedocumenteerd model dat door de jaren heen meegroeit met de organisatie. De implementatie ervan geschiedt steeds op het platform dat op dat moment gewenst is. Het idee achter MDA is vertrouwd: eerst wordt het systeem op een hoog abstractieniveau gemodelleerd, vervolgens wordt het systeem volledig gespecificeerd in modellen, waaruit vervolgens een compleet werkende applicatie wordt gegenereerd.

MDA onderkent drie niveaus:

- een platformonafhankelijk model (PIM – Platform Independent Model);
- een platformspecifiek model (PSM – Platform Specific Model);
- het implementatiemodel.

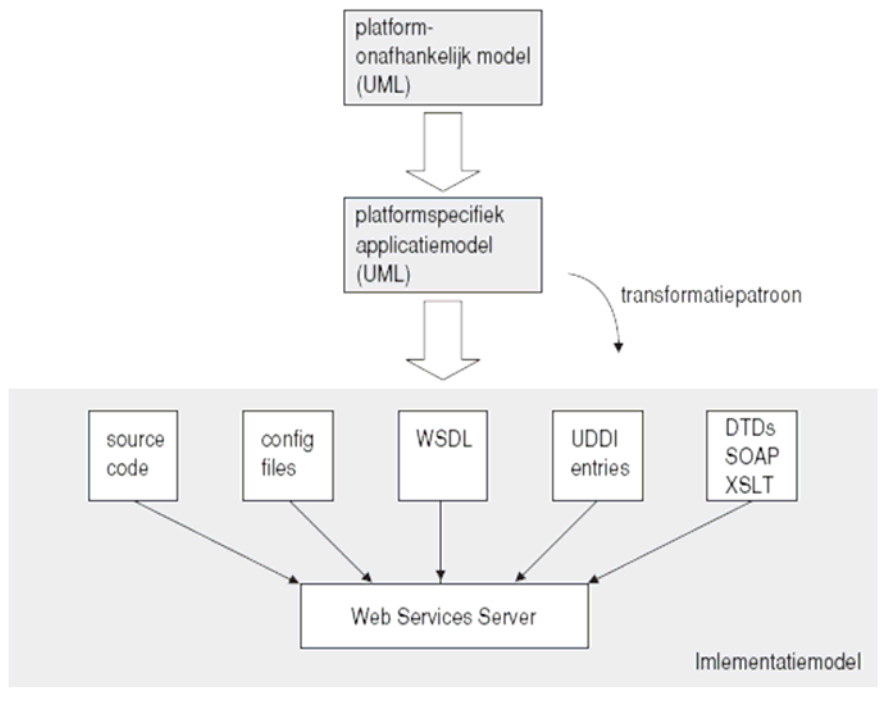
Het platformonafhankelijke model beschrijft de functionaliteit en het gedrag (business rules) van het informatiesysteem. Het platformspecifieke model beschrijft het systeem voor een specifiek doelplatform, bijvoorbeeld voor J2EE of Microsoft .NET, of een combinatie van beide. Het implementatiemodel omvat alle code die nodig is om de applicatie te executeren op het doelplatform. De code wordt gegenereerd vanuit het platformspecifieke model. De modellen binnen MDA worden opgesteld in de Unified Modelling Language (UML)-notatie.

De transformaties van platformonafhankelijk model naar platformspecifiek model, en van platformspecifiek model naar code, worden bij voorkeur door geautomatiseerde gereedschappen uitgevoerd. MDA definieert vier soorten transformaties:

1. Een persoon voert de transformatie handmatig uit (ad hoc).
2. Een persoon voert de transformatie handmatig uit gebruikmakend van bewezen transformatiepatronen.
3. Een geautomatiseerd gereedschap voert de transformatie uit aan de hand van transformatiepatronen en produceert een initieel model dat vervolgens handmatig wordt gecompleteerd.
4. De gehele transformatie wordt automatisch door een MDA-tool uitgevoerd.

Figuur 6 geeft schematisch de aanpak bij de toepassing van MDA weer.

Een transformatiepatroon biedt een ‘voorgebakken’ oplossing voor een bepaald probleem naar een specifiek technologieafhankelijk platform. De eis ‘persistentie’ bijvoorbeeld kan vanuit een platformspecifiek model naar Java worden omgezet door middel van een transformatiepatroon richting Entity Beans of bijvoorbeeld naar Java Data Objects. Een MDA-tool biedt voor deze transformatie twee patronen aan; de keuze is aan de gebruiker van het MDA-tool.



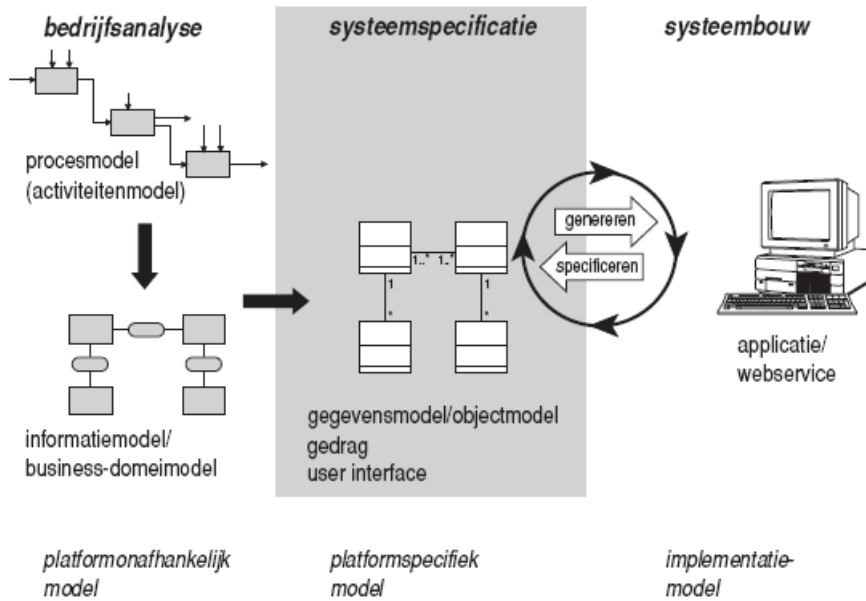
Figuur 6 Toepassing van MDA voor het ontwikkelen van een webservice

De concurrentie tussen de MDA-tools zal zich toespitsen op wie de slimste transformatiepatronen heeft. Daarnaast zal de keuze voor een MDA-tool worden beïnvloed door de hoeveelheid aanwezige patronen, de aanpasbaarheid van de patronen en de mogelijkheid van het toevoegen van zelfgedefinieerde patronen. Bovendien wordt de keuze bepaald door de mate waarin het MDA-tool aansluit op de standaardisatie van de talen waarmee de transformatiepatronen worden beschreven. Immers, investeren in een specifieke patroontaal kan ongewenst zijn in verband met een mogelijke (lock-in) afhankelijkheid van de leverancier.

Figuur 7 illustreert de afbeelding van MAD op MDA.

Binnen MDA is gekozen voor een objectgeoriënteerde nomenclatuur; MAD voorziet ook in een ‘klassieke’ gegevensgeoriënteerde benaming. De concepten voor het modelleren en genereren van applicaties vertonen een grote mate van overeenkomst. Zowel bij MDA als bij MAD is een model dat de dingen uit de werkelijkheid, hun kenmerken en hun gedrag beschrijft, het uitgangspunt voor de ontwikkeling van software. Bij MDA is dit het platform-onafhankelijke model; bij MAD wordt dit model weergegeven als een informatiemodel of – bij een objectgeoriënteerde aanpak – een business-domeinmodel. Bij MDA wordt het platformonafhankelijke model getransformeerd naar een platformspecifiek model. Vertaald naar MAD: het informatiemodel wordt omgezet naar een relationeel gegevensmodel of – bij een objectgeoriënteerde aanpak – het business-domeinmodel wordt getransformeerd naar een objectmodel (klassendiagram). De functionaliteit en het gedrag van het systeem worden expliciet gemodelleerd met event-getriggerde regels. Vervolgens worden vanuit deze modellen alle code en tabellen gegenereerd die nodig zijn om de

applicatie te kunnen executeren op het doelplatform. Hiermee wordt invulling gegeven aan het (MDA-) implementatiemodel.



Figuur 7 MAD in relatie tot MDA

De volgende tabel geeft de relatie weer tussen de terminologie van de MDA, de gegevensgeoriënteerde benadering en de objectgeoriënteerde benadering (gebaseerd op de 3-schema-architectuur).

	Model Driven Architecture	Gegevensgeoriënteerde benadering	Objectgeoriënteerde benadering
Conceptueel	Platformonafhankelijk model	Informatiemodel	Businessdomeinmodel
Extern schema	Platformspecifiek model	Gegevensmodel	Objectmodel
Intern schema	Implementatiemodel	Databaseschema	Classes/ Databaseschema

1.6 Samenvatting

In dit hoofdstuk zijn de basiselementen van MAD belicht:

- De Software Engineering-benadering is gebaseerd op modelleren.
- Applicatiegeneratoren automatiseren ontwerp- en bouwactiviteiten van het ontwikkelproces. Zij hebben ook een stroomlijnend effect op de analyse- en specificatieactiviteiten, aangezien de informatieanalist zich in eerste instantie kan concentreren op het verkrijgen van die modellen die leiden tot een eerste werkend prototype.
- Incrementeel en iteratief ontwikkelen staat een parallelle ontwikkeling toe en versnelt de systeemoplevering, waarbij de specificaties en het systeem tijdens een aantal iteraties worden verkregen en verfijnd.
- Joint Application Design beoogt een versnelling en verbetering van het specificatieproces door een intensieve betrokkenheid van de juiste personen.
- Een applicatie-templates kan worden ingezet om een vliegende start te geven aan analyse en specificatie. Het initieel gegenereerde informatiesysteem doet dienst als een eerste prototype ter verkrijging van een 'dedicated' maatwerksysteem.
- Model Driven Architecture – een initiatief van de Object Management Group – definieert een set van UML-modellen, waarbij de transformatie tussen de modellen en de transformatie van modellen naar code wordt ondersteund door geautomatiseerde gereedschappen. Dit initiatief heeft in de wereld van objectoriëntatie een nieuwe impuls gegeven aan de inzet van applicatiegeneratoren en de toepassing van een modelgedreven benadering, waarbij applicaties vanuit een modelmatig gepopuleerde repository worden gegenereerd.

2 De MAD-modellencyclus

Dit hoofdstuk behandelt de MAD-modellencyclus. Het beschrijft de voor MAD relevante modellen en hun onderlinge samenhang.

2.1 De modellencyclus

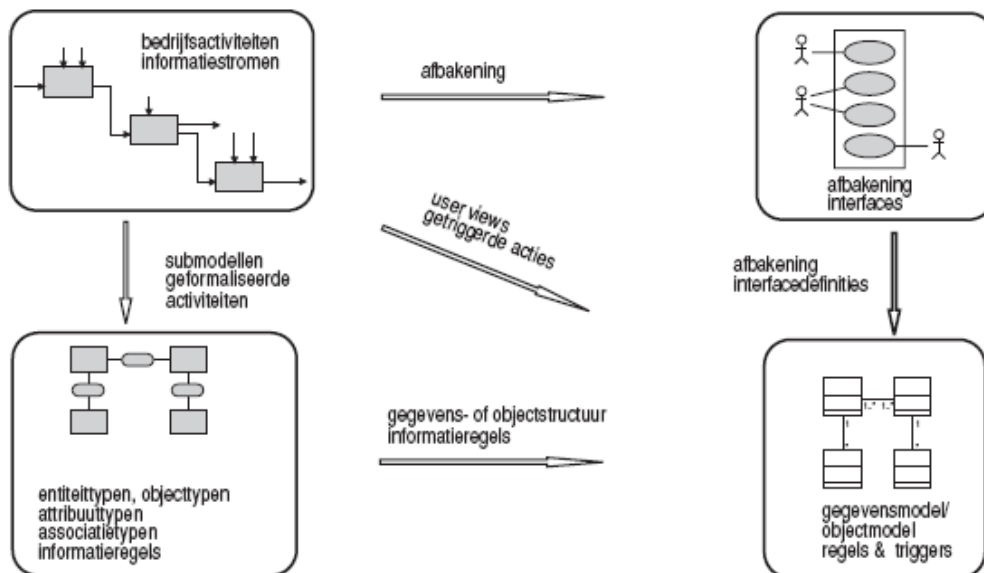
Een model is een vereenvoudigde weergave van de werkelijkheid, waarin een of meer aspecten worden belicht. Bij modelleren is het belangrijk dat men zich realiseert en dat men uit elkaar houdt:

- welke wereld wordt beschouwd (is het universum van discussie of het geautomatiseerde informatiesysteem het onderwerp van modelleren);
- welk aspect in het model wordt benadrukt (activiteiten of informatie).

Binnen MAD worden twee categorieën van modellen onderkend:

- het analysemodel – als resultaat van een bedrijfsanalyse – bestaande uit een bedrijfsactiviteitenmodel en een informatiemodel;
- het specificatiemodel – als resultaat van een systeemspecificatie – bestaande uit een systeemafbakening en een gegevensmodel of objectmodel met ingesloten functionaliteitspecificatie.

Figuur 8 toont de diverse modeltypen en hun onderlinge samenhang. In de paragrafen hieronder worden deze modellen nader beschouwd.



Figuur 8 De modellencyclus

2.2 Het bedrijfsactiviteitenmodel

Doel

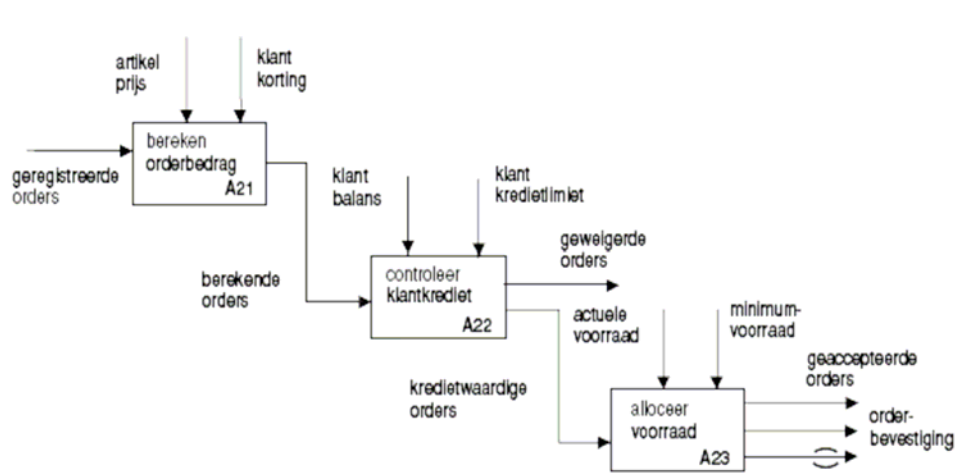
Het modelleren van bedrijfsactiviteiten wordt toegepast voor het analyseren en documenteren van de bedrijfsactiviteiten. Het doel is te begrijpen hoe het bedrijf of bedrijfsonderdeel werkt en welke activiteiten eventueel kunnen worden ondersteund met behulp van een geautomatiseerd informatiesysteem. De resultaten worden vastgelegd in het activiteitenmodel. Dit model toont de bedrijfsactiviteiten en hun onderlinge samenhang met betrekking tot informatie-uitwisseling. Het bedrijfsmodel vormt een goed uitgangspunt voor informatiemodellering.

Activiteitenmodel

Het activiteitenmodel bestaat uit twee delen:

- een beschrijving van de bedrijfsactiviteiten en hun informatie-uitwisseling;
- een activiteitendiagram.

Figuur 9 toont een voorbeeld van een activiteitendiagram.



Figuur 9 Voorbeeld van een activiteitendiagram

Methoden

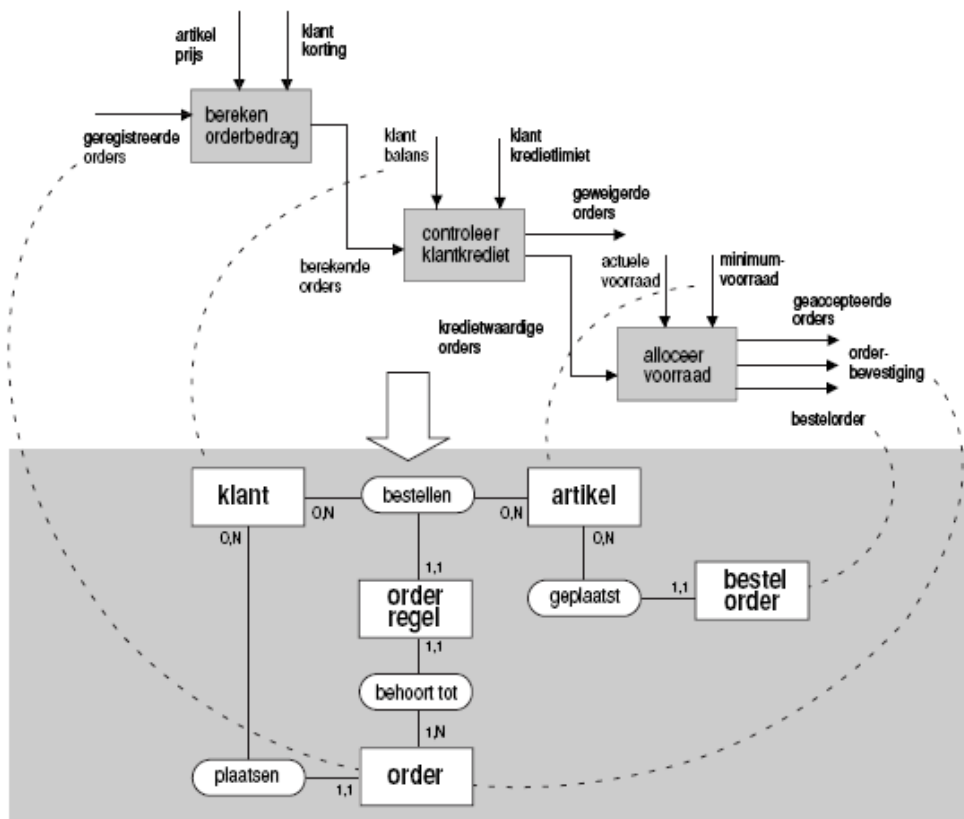
De methode Actimod (ACTivity MODelling) ondersteunt het modelleren van de bedrijfsactiviteiten. Deze methode hanteert de IDEF0-techniek. Actimod is afgeleid van SADT – Structured Analysis and Design Technique [Marca]. Andere methoden, zoals ISAC (A-schema's) en Yourdon of Hatley/Pirbhai (de 'dataflow diagrams') kunnen hiervoor ook worden toegepast.

Van activiteitenmodel naar specificaties

Het activiteitenmodel – als beschrijving van de bedrijfsactiviteiten – vormt al de basis voor het opstellen van enige specificaties van het informatiesysteem. Als zodanig kan het activiteitenmodel worden gebruikt voor de volgende specificaties.

- Het vaststellen van de systeemomvang. De resultaten worden vastgelegd in een contextdiagram.
- De in het activiteitenmodel vastgelegde informatiestromen zijn berichten over entiteitstypen. Deze entiteitstypen worden gegroepeerd in informatie(sub)modellen.
- Bedrijfsactiviteitenmodellen die volledig formaliseerbaar zijn worden gespecificeerd als informatieregels.
- Voor elke bedrijfsactiviteit (gebruikersgroep) kan worden vastgesteld welke informatie benodigd is. Dit vormt de basis voor de externe schema's.

Figuur 10 illustreert het verband tussen het bedrijfsactiviteitenmodel en het informatie-model.



Figuur 10 Van bedrijfsactiviteitenmodel naar informatiemodel

Het activiteitenmodel toont de inkomende en uitgaande informatiestromen van de bedrijfsactiviteiten. Deze stromen zijn berichten over gebeurtenissen en entiteitstypen. Aan

de inhoud van deze stromen kunnen entiteitstypen en attribuuttypen worden ontleend, die vervolgens worden gestructureerd in een informatiestructuurdiagram.

2.3 Het informatiemodel

Doel

Informatiemodellering wordt gebruikt voor het analyseren en documenteren van de informatiebehoefte van een bedrijf. De resultaten worden vastgelegd in het informatiemodel. Het informatiemodel beschrijft de entiteitstypen, attribuuttypen en informatieregels. Het informatiemodel dient als een architectuur voor informatiesystemen en is dus een belangrijke component voor informatiesysteemontwikkeling en -onderhoud. Informatiemodellering is een bedrijfsgerichte analyseactiviteit.

De invalshoek van informatiemodellering verschilt van die van gegevensmodellering. Informatiemodellering belicht de bedrijfswerkelijkheid. Een informatiemodel toont de entiteitstypen (mensen, dingen, gebeurtenissen) en hun karakteristieken (attribuuttypen en regels) die door organisaties relevant worden geacht.

Er zijn grote overeenkomsten tussen het informatiemodel en het business-domeinmodel. Dat is niet zo verwonderlijk wanneer men in oenschouw neemt dat beide de dingen uit de werkelijkheid, hun kenmerken en gedrag beschrijven. De verschillen tussen beide uiten zich meer in het modelleren van het gedrag. Bij een informatiemodel wordt dat beschreven in informatieregels; bij een businessdomeinmodel wordt het gedrag expliciet mee gemodelleerd in de vorm van ingesloten functionaliteit (operations).

Ook de notatievorm van een informatiemodel en een business-domeinmodel vertoont overeenkomsten; beide zijn gebaseerd op de Entity-Relationship-methode. Toch is het een kwestie van modelleren: het onderscheid tussen een gegevensobject met ingesloten functionaliteit in de vorm van informatieregels en een business-object met ingesloten functionaliteit in de vorm van operations is vaag.

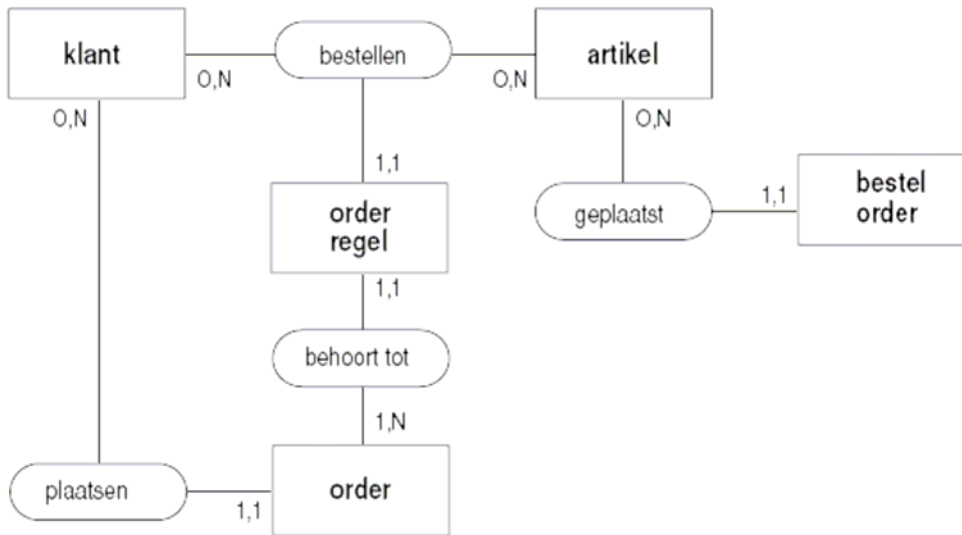
Informatiemodel

Het informatiemodel bestaat uit twee delen:

- het informatiestructuurmodel;
- de informatieregels (bedrijfsregels).

Informatiestructuurmodel

Het informatiestructuurmodel bestaat doorgaans uit een informatiestructuurdiagram (ook wel entiteitstructuurdiagram genoemd) en een beschrijving van de entiteitstypen. Het informatiestructuurdiagram toont de entiteitstypen en hun onderlinge associaties. Figuur 11 geeft hiervan een voorbeeld. De beschrijving van de entiteitstypen bestaat uit de naam van het entiteitstype, de identificerende attributen en een lijstje van attributen. Het onderstaande tekstkader geeft de beschrijving van de entiteitstypen uit figuur 11.



Figuur 11 Voorbeeld van een informatiestructuurdiagram

Entity type: Klant	Entity type: Order
Identifier: klantcode	Identifier: ordernummer
Description: klantcode	Description: ordernummer
naam	orderstatus
adres	orderdatum
balans	orderbedrag
kredietlimiet	
korting	Entity type: Orderregel
	Identifier: ordernummer
Entity type: Artikel	Identifier: regelnummer
Identifier: artikelnummer	Description: regelnummer
Description: artikelnummer	besteld aantal
omschrijving	
prijs	Entity type: Bestelorder
minimum-voorraad-niveau	Identifier: leverancierordernummer
voorraadbalans	Description: leverancierordernummer
	aantal

Informatieregels

We onderscheiden de volgende categorieën informatieregels (bedrijfsregels):

- identificatieregels;
- verplichte attributen;
- validatieregels;
- afhankelijkheidsregels;
- afleidingsregels;
- transitieregels.

Een identificatieregels geeft aan welk attribuut of welke combinatie van attributen identificerend is voor een entiteit. In het voorbeeld hierboven zijn de identificatieregels een onderdeel van de entiteitstypenbeschrijvingen. Voorbeeld: een klantcode identificeert een klant.

Een verplichte-attributenregel definieert welke attributen bekend moeten zijn in het geval dat de entiteit bekend is. Voorbeeld: wanneer een klant bekend is, moeten zijn klantcode, naam en adres bekend zijn.

Een validatieregels definieert de toegestane waarden voor een bepaald attribuut.

Voorbeelden:

- de orderstatus is ‘geregistreerd’, ‘berekend’, ‘geweigerd’, ‘kredietwaardig’ of ‘geaccepteerd’;
- kredietlimiet ≥ 0 ;
- toegestane waarden voor korting zijn ‘0’, ‘10’, ‘20’ en ‘30’.

Een afhankelijkheidsregel beschrijft onderlinge afhankelijkheden tussen attributen van één entiteit of tussen attributen van verschillende entiteiten. In het voorbeeld van figuur 11 bestaat er een afhankelijkheid tussen de associaties ‘bestellen’, ‘plaatsen’ en ‘behoort tot’.

Een afleidingsregel definieert dat de waarde van een attribuut kan worden afgeleid uit de waarden van andere attributen. Voorbeeld: het orderbedrag wordt berekend als besteld aantal * artikelprijs * (1 – korting/100).

Een transitieregel beschrijft de toegestane veranderingsregels of status-overgangen.

Voorbeelden van toegestane transities voor orderstatus zijn:

- van ‘geregistreerd’ naar ‘berekend’;
- van ‘berekend’ naar ‘kredietwaardig’;
- van ‘berekend’ naar ‘geweigerd’;
- van ‘kredietwaardig’ naar ‘geaccepteerd’.

Methoden

Voor informatiemodellering is een aantal methoden beschikbaar. We noemen hier de volgende:

- Entity-Relationship;
- NIAM;
- Infomod;
- Unified Modeling Language.

De Entity-Relationship-methode is oorspronkelijk ontwikkeld door P.P. Chen. Later is het door hem ontwikkelde basismodel uitgebreid met toevoegingen zoals meerwaardige attributen, bestaansafhankelijke entiteitstypen, aggregatie, subtype en generalisatie [Laagland]. De Entity-Relationship-methode is de meest wijdverbreide methode ter wereld, mede doordat er vele CASE-tools zijn die haar ondersteunen. Hoofdstuk 6 beschrijft de Entity-Relationship-methode.

NIAM – Nijssen Informatie Analyse Methode – is gebaseerd op natuurlijke taal, waarbij de werkelijkheid wordt beschreven door middel van een set van elementaire beweringen. Deze beweringen kunnen voor een groot gedeelte grafisch worden weergegeven. NIAM kent twee varianten: binair [Wintraecken] en N-air [de Rooij]. Het informatiestructuurdiagram kent ook een syntaxis voor het vastleggen van enkele informatieregels.

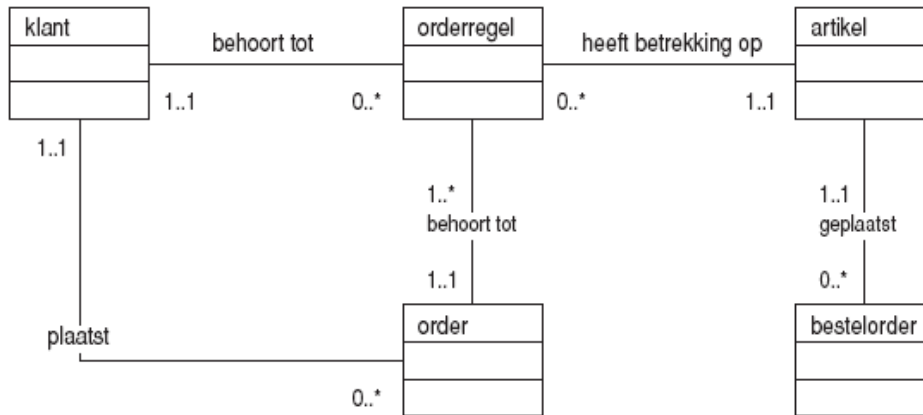
FCO-IM – Volledig Communicatiegeoriënteerde informatiemodellering – bouwt voort op de principes en schematechnieken van NIAM. Het is een methode voor informatie-modellering en voor relationeel gegevensontwerp [Bakema]. In deze lijn past ook de methode EXPO – een methode voor een objectgeoriënteerde domeinanalyse. EXPO (Expression based object modelling) is een systematische aanpak voor het ontwerpen van klassendiagrammen. De aanpak voor het structureren van informatie is – vergelijkbaar met NIAM en FCO-IM – gebaseerd op feiten uit de werkelijkheid.

Infomod is een bij Philips ontwikkelde methode voor informatiemodellering. Naast het informatiestructuurdiagram kent deze methode ook een uitgebreide syntaxis voor het beschrijven van het informatiemodel, inclusief de informatieregels [Griethuysen]. De kracht van Infomod ligt enerzijds in de communicatie van de informatieanalist met de gebruiker en anderzijds in het formele karakter van het op te leveren model.

Unified Modeling Language (UML) is een grafische taal om de structuur van informatie en objectgeoriënteerde programma's vast te leggen. Centraal in UML staat het klassendiagram. Het is een model van klassen met attributen en operations, die door middel van associaties met elkaar in verbinding staan. In tegenstelling tot Infomod, NIAM en EXPO, is UML voornamelijk een notatiewijze en definieert het de syntaxis van verschillende diagrammen. UML is inmiddels geaccepteerd als een industriestandaard.

UML maakt onderscheid tussen een informatiediagram en een klassendiagram. Voor beide diagrammen wordt dezelfde notatiewijze gehanteerd. Figuur 12 illustreert het informatiemodel uit figuur 11 in de UML-notatie, waarbij alleen het gegevensaspect (zonder attributen) is gemodelleerd.

De voorbeelden van informatiemodellen in dit boek zijn opgesteld in de notatie van de Entity-Relationship-methode volgens de International Organization for Standardization (ISO) [ISO]. Deze methode wordt ook nader beschreven in hoofdstuk 6. Van enkele voorbeelden wordt ook de UML-variant gepresenteerd.



Figuur 12 Voorbeeld van een informatiediagram in UML-notatie

2.4 Systeemaafbakening

De eerste specificatieactiviteit is het vaststellen van de systeemaafbakening. Aan de hand van het activiteitenmodel wordt bepaald welke bedrijfsactiviteiten voor automatisering in aanmerking komen. Hiertoe wordt binnen de grenzen van het applicatiegebied bepaald welk gedeelte van de bedrijfsactiviteiten wordt ondersteund of uitgevoerd door het geautomatiseerde informatiesysteem.

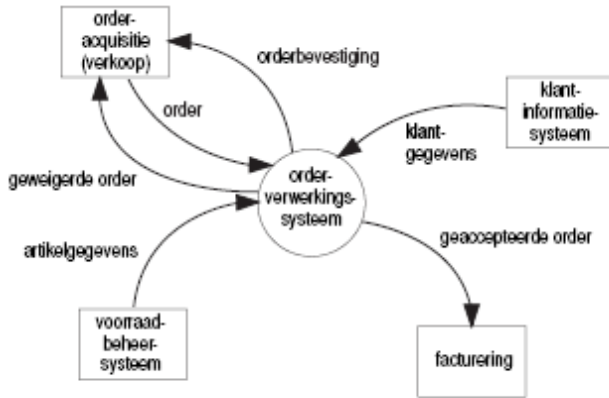
Vervolgens worden de grenzen van het informatiemodel vastgesteld en worden de bedrijfsactiviteiten bepaald die gaan communiceren met het systeem. Tevens worden de ingaande en uitgaande informatiestromen gedefinieerd.

Het resultaat wordt gedocumenteerd in een systeemcontextdiagram (omgevingsmodel). Dit diagram wordt afgeleid van het activiteitendiagram en toont het systeem als ‘black box’ en de interfaces met andere systemen en hun gebruikers.

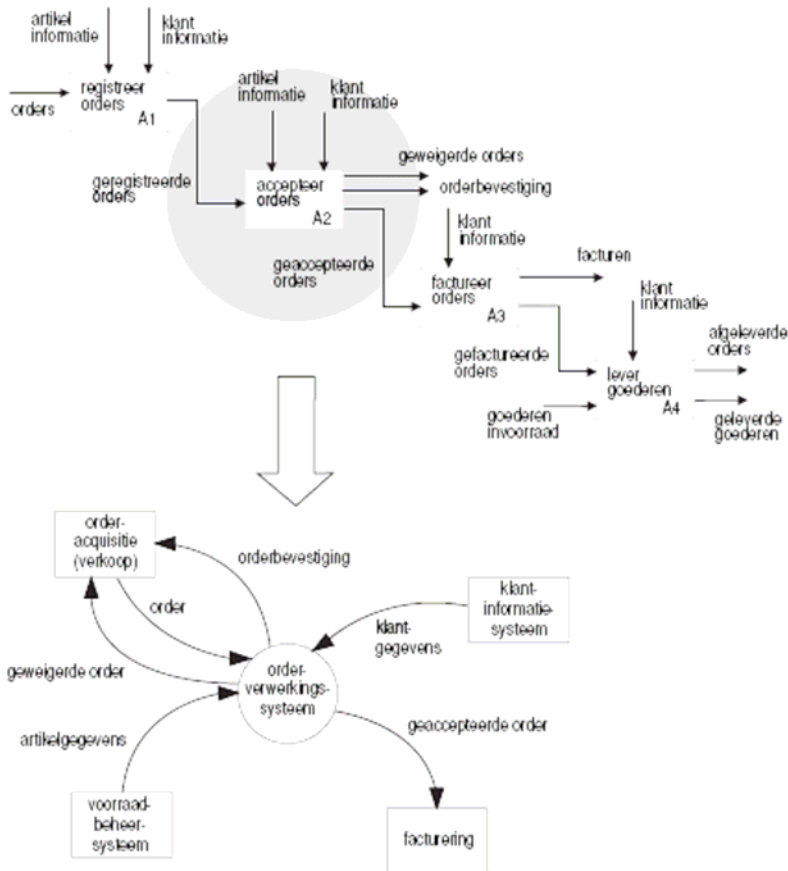
Bedrijfsactiviteiten die worden ondersteund of uitgevoerd door het geautomatiseerde informatiesysteem zijn onderdeel van het systeem. Het te ontwikkelen informatiesysteem omvat en ondersteunt deze bedrijfsactiviteiten. Zij vormen de scope van het systeem.

Figuur 13 toont een voorbeeld van een systeemcontextdiagram.

Figuur 14 illustreert de overgang van het bedrijfsactiviteitenmodel naar het contextdiagram. De te automatiseren activiteiten (in dit voorbeeld slechts één) vormen samen het te bouwen systeem. De informatiestromen van deze activiteiten met activiteiten die geen deel uitmaken van het systeem, verschijnen als gegevensstromen op het contextdiagram. Deze gegevensstromen representeren de interfaces tussen het te bouwen informatiesysteem en de buitenwereld (‘terminators’, ‘external entities’).



Figuur 13 Voorbeeld van een contextdiagram

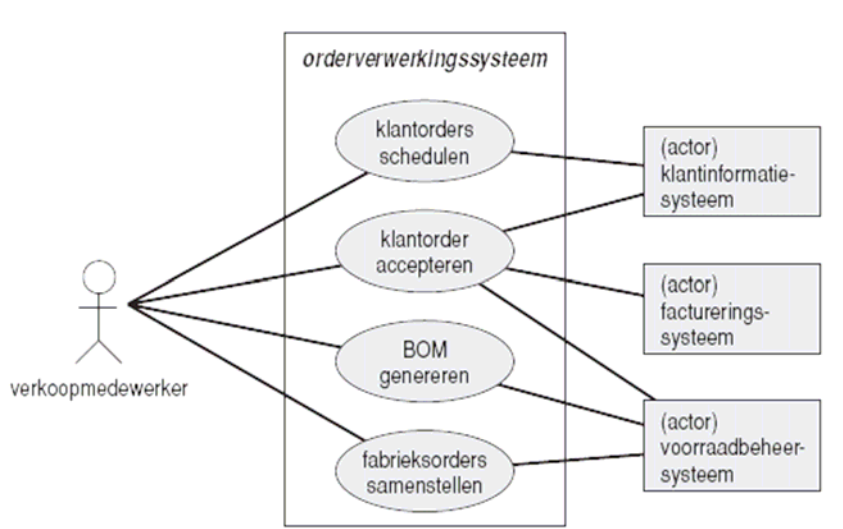


Figuur 14 Bepaling van de systeemgrenzen

Methoden

De contextdiagrammen binnen de methoden Yourdon en Hatley/Pirbhai – ‘context data flow diagrams’ – zijn veel toegepaste technieken voor het documenteren van de systeemafbakening. Echter, ‘vrije’ plaatjes voldoen ook ter illustratie van de systeemafbakening. Binnen de MAD-modellencyclus wordt verder geen aandacht besteed aan het modelleren van systeemfuncties met behulp van procesmodellen (de ‘data flow diagrams’). Het modelleren van systeemfunctionaliteit wordt op een andere wijze afgedekt (zie paragraaf 2.6).

Binnen UML wordt de systeemfunctionaliteit gemodelleerd door middel van use cases. Een use case beschrijft de functionaliteit van het systeem vanuit de gebruikersgroepen. Figuur 15 geeft het contextdiagram uit figuur 13 weer als een use case in de UML-notatie.



Figuur 15 Voorbeeld van een omgevingsmodel (use case) in UML-notatie

2.5 Het gegevensmodel

Doel

Gegevensmodellering wordt gebruikt bij het definiëren van de wijze waarop de in het informatiemodel vastgelegde informatiebehoefte wordt gestructureerd en gerepresenteerd door middel van gegevens in de database en naar de gebruikers.

Het is een systeemgerichte specificatieactiviteit en belicht het gegevensaspect van het (geautomatiseerde) informatiesysteem. De resultaten worden vastgelegd in het gegevensmodel. Dit model belicht het externe (data)niveau. In het gegevensmodel worden de gegevensstructuur van de gegevensverzamelingen en de externe representatie van de gegevens gedefinieerd.

Gegevensmodellering betreft het externe (gegevens)aspect van het geautomatiseerde informatiesysteem. Het onderwerp van modellering is het geautomatiseerde informatiesysteem: welke gegevens, in welke vorm, worden opgeslagen in het systeem.

Methoden

Er zijn verschillende typen gegevensmodellen. In dit boek richten wij ons op het relationele model [Date], omdat dit modeltype algemeen bekend is en omdat het relationele model ten grondslag ligt aan de thans populaire databasemanagementsysteem. Het Bachman-diagram is gekozen als grafische representatie van het relationele model.

Terminologie

De hieronder vermelde begripsomschrijvingen zijn niet de precieze definities van de begrippen zoals die bij het relationele model worden gebruikt. Een nauwkeuriger definitie is opgenomen in hoofdstuk 7. Voorlopig gebruiken we echter deze omschrijvingen.

Tabel: de weergave van een entiteitstype.

Attribuut van een tabel: de weergave van een attribuuttype.

Tupel: de weergave van een entiteit-occurrence.

Primaire sleutel van een tabel: de weergave van een identificatie.

Vreemde sleutel van een tabel: de weergave van een associatie tussen entiteitstypen.

Normaliseren en informatie modelleren

Normaliseren is een begrip dat onlosmakelijk verbonden is met het relationele model, zoals dat door Edward Codd in de beginjaren '70 is geformuleerd. Het doel van normaliseren is om door middel van een beperkt aantal stappen (de 'normalisatiestappen') te komen tot een gegevensstructuur, een model voor een database, waarin alleen maar enkelvoudige, elementaire gegevens voorkomen (dus geen 'repeating groups' en geen 'afleidbare gegevens') en waarin bovendien elk gegeven maar één keer voorkomt. De achterliggende gedachte was dat, als zich in de werkelijkheid één feit voordoet, dat ook maar op één plek in de database resulteert in een toevoeging, een wijziging of een verwijdering. Men zegt ook wel dat een genormaliseerd relationeel gegevensmodel 'update-geoptimaliseerd' is (het voorkomt 'updateanomalieën').

Tegenwoordig, met de huidige kennis van informatiemodellen, zouden we zeggen dat we ervoor moeten zorgen dat alle attributen bij de juiste entiteitstypen moeten worden gemodelleerd. Als we een correct informatiemodel hebben, dat we met enkele, haast mechanistische, transformaties omzetten naar een relationeel model, dan is dat model al genormaliseerd. In die zin is normaliseren 'uit de tijd'.

Als we de twee wegen (informatie modelleren en normaliseren) om tot een relationeel model te komen even zwart-wit tegenover elkaar stellen, zien we het volgende verschil:

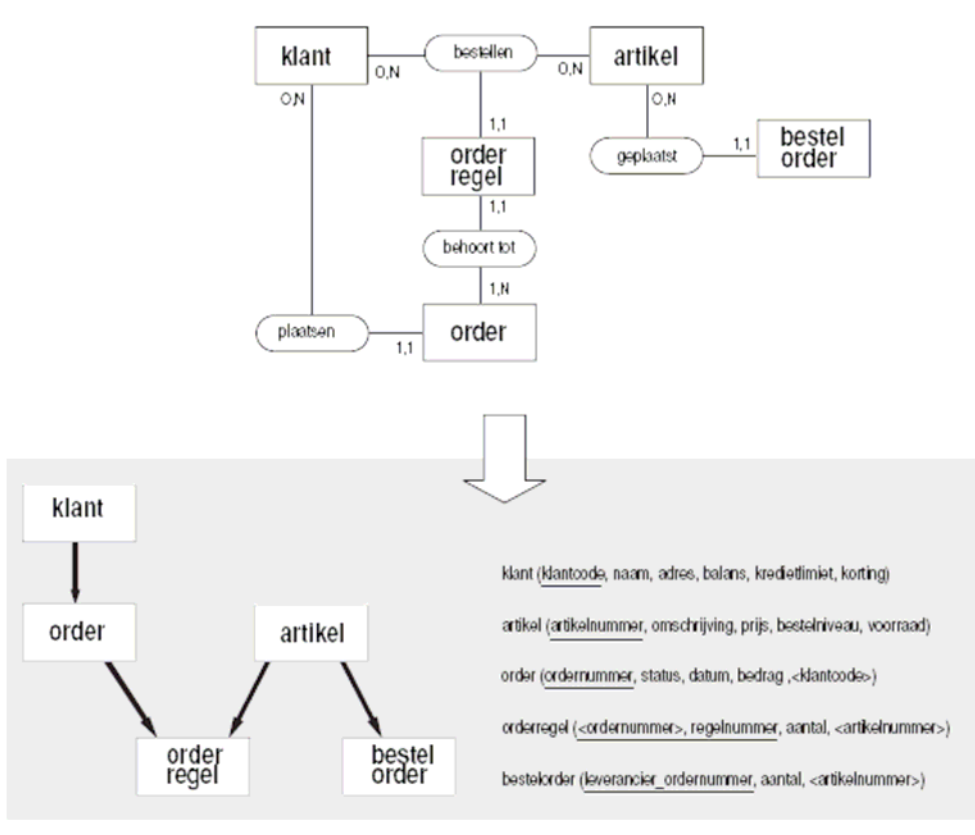
- Met informatiemodellering poogt men, door de beschouwde werkelijkheid (het universum van discussie) te bestuderen, te komen tot een correct informatiemodel, dat

bij toepassing van enkele voorgeschreven transformaties resulteert in een genormaliseerd relationeel gegevensmodel.

- Met normaliseren gaat men uit van een ‘ongestructureerd, willekeurig, fout’ gegevensmodel en poogt dat in een aantal voorgeschreven stappen te structureren door alsnog vragen te stellen over de beschouwde werkelijkheid (het universum van discussie).

In de praktijk zal het wel minder zwart-wit zijn: het uitgangsmodel heeft normaliter toch wel enige structuur. Maar we kunnen wel zeggen dat normaliseren meer een controle achteraf is dan een ontwerpactiviteit.

Het omzetten van een informatiemodel naar een relationeel gegevensmodel is een mechanistische aangelegenheid. Figuur 16 illustreert deze overgang. Als eerste stap wordt elk entiteitstype gerepresenteerd door een tabel en wordt elk attribuuttype een kolom (domein). Vervolgens worden de associaties tussen de entiteitstypen weergegeven door vreemde sleutels. Meer-op-meer-associaties (M:M-associaties) en ternaire associaties worden afgebroken tot binaire associaties.

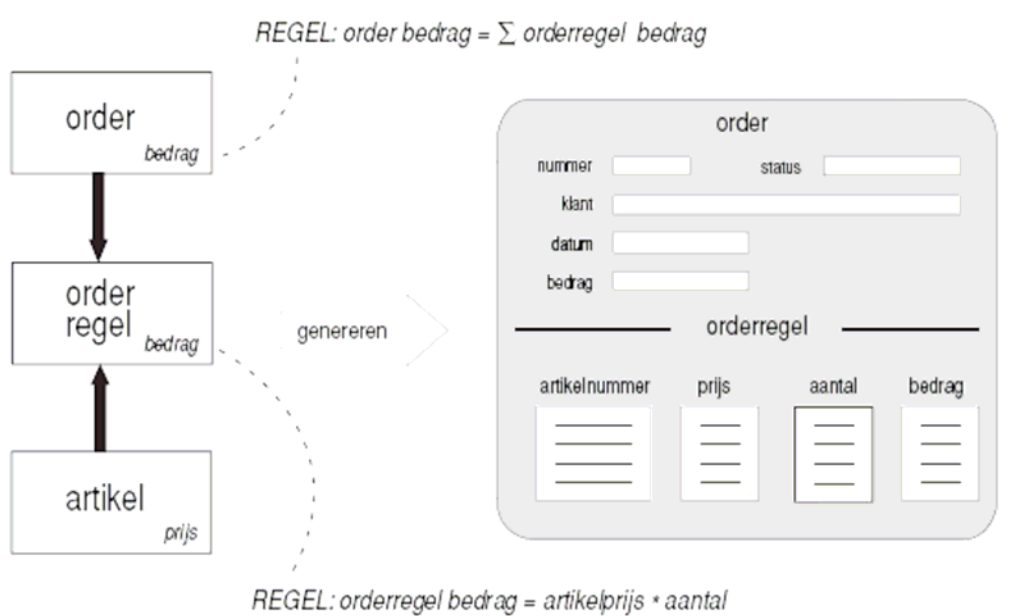


Figuur 16 Van een informatiemodel naar een relationeel gegevensmodel

Hoofdstuk 7 beschrijft enkele begrippen uit de relationele theorie, belicht normalisatie en bevat enkele handvatten voor het omzetten van een informatiemodel naar een relationeel gegevensmodel.

Procesgegevens

Strikt volgens de relationele theorie voor relationele databases zijn procesgegevens niet toegestaan. Procesgegevens zijn gegevens die kunnen worden afgeleid uit andere gegevens. Echter, in de MAD-benadering worden deze, samen met hun afleidingsregels, expliciet gemodelleerd. Figuur 17 illustreert dit. In dit voorbeeld zijn orderbedrag en orderregelbedrag beide procesgegevens. Door deze in het gegevensmodel op te nemen verschijnen ze op de gegenereerde schermen. Hun waarden worden berekend met behulp van de afleidingsregels. Procesgegevens kunnen als databasevelden of als 'virtuele' velden worden geïmplementeerd.

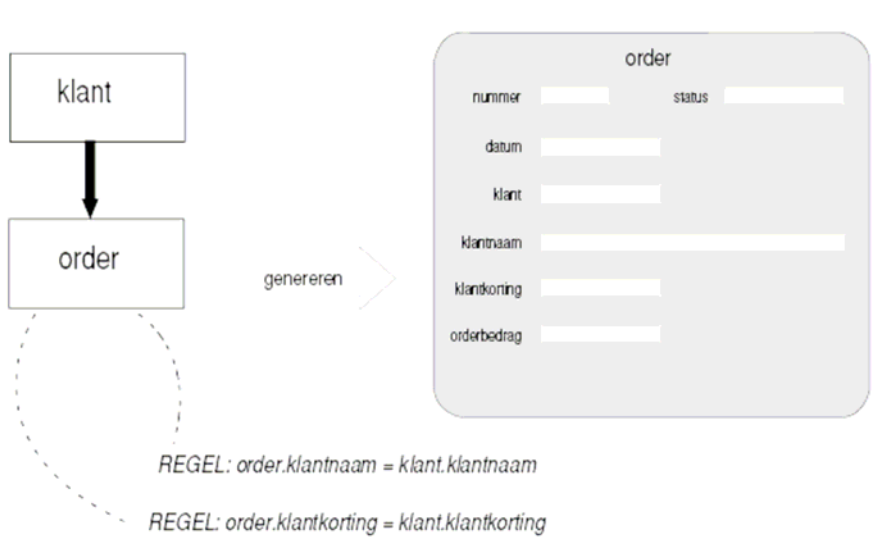


Figuur 17 Procesgegevens

Denormalisatie

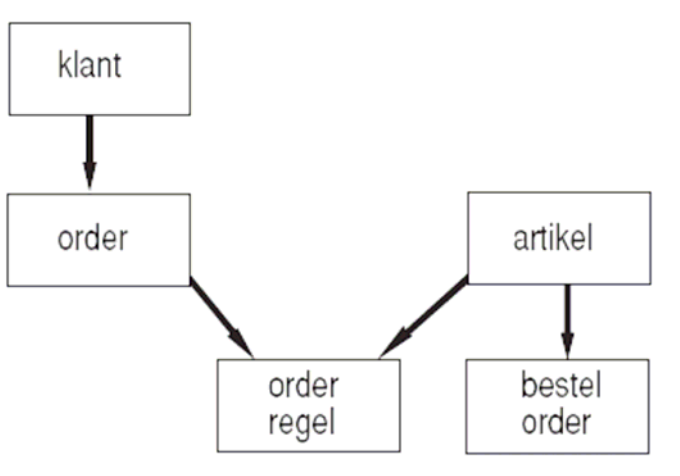
Volgens de theorie moeten relationele gegevensmodellen genormaliseerd zijn. In de praktijk worden sommige tabellen in een gegevensmodel bewust gedенormaliseerd, dat wil zeggen teruggebracht naar een lagere normaalvorm dan BCNF (de Boyce-Codd-normaalvorm, zie paragraaf 7.2). De reden hiervoor is meestal een beoogde performanceverbetering. Hiermee wordt tegelijkertijd een inconsistentieprobleem geïntroduceerd. Dit is echter op te lossen door een afleidingsregel te definiëren. Daarmee is de denormalisatie op specificatieniveau gecontroleerd.

Figuur 18 geeft hiervan een voorbeeld. In dit voorbeeld zijn enkele attributen van klant (klantnaam en klantkorting) naast de vreemde sleutel 'klantcode' opgenomen bij de order. Performance kan hiervoor een reden zijn, maar ook het feit dat de eindgebruiker dit zo wil zien (view). Door regels toe te voegen aan de velden worden de actuele waarden vanuit de tabel klant opgehaald.



Figuur 18 Denormaliseren

Figuur 19 toont een voorbeeld van een grafische weergave van een gegevensmodel, ontleend aan figuur 11.



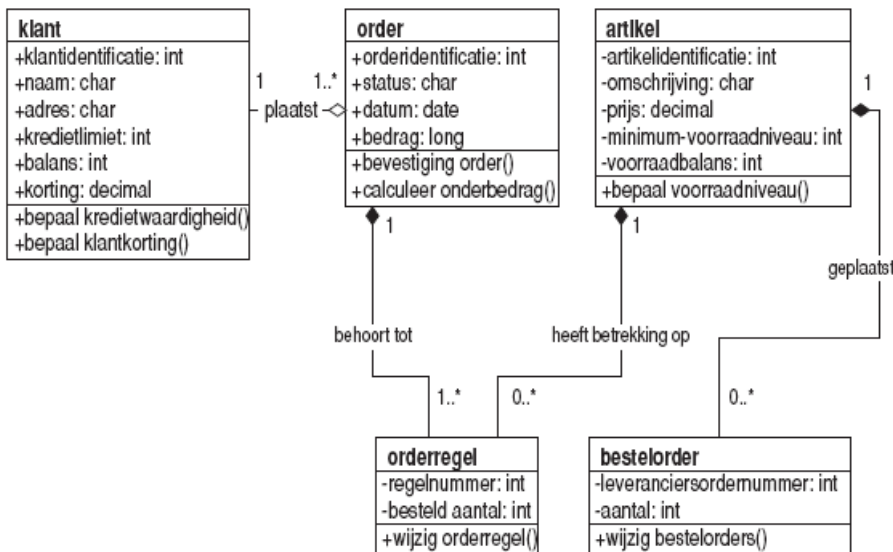
Figuur 19 Voorbeeld van een grafische weergave van een gegevensmodel in de Bachman-notatie

Het onderstaande tekstkader presenteert de tekstuele beschrijving van het betreffende model.

Klant (klantcode, naam, adres, balans, kredietlimiet, korting)
 Artikel (artikelnummer, omschrijving, prijs, minimum-voorraadniveau, voorraadbalans)
 Order (ordernummer, orderstatus, orderdatum, orderbedrag, <klantcode>)
 Orderregel (<ordernummer>, regelnummer, besteld aantal, <artikelnummer>)
 Bestelorder (leverancierordernummer, aantal, <artikelnummer>)

Primaire sleutels zijn onderstreept afgedrukt. Vreemde sleutels staan tussen punthaken.

Figuur 20 geeft het gegevensmodel uit figuur 19 weer als een objectmodel in de UML-notatie.



Figuur 20 Voorbeeld van een objectmodel (klassendiagram) in UML-notatie

2.6 Functionaliteit

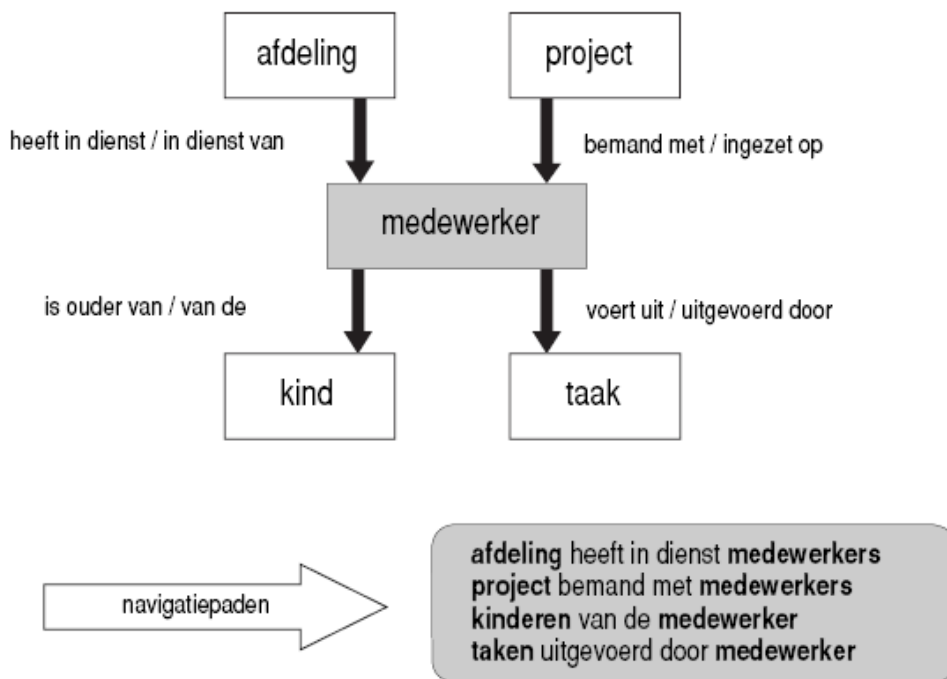
Gebruik tabellen/objecten

De meeste applicatiegeneratoren genereren van de gegevensstructuur of vanuit de objectenstructuur functies voor het opvragen, toevoegen, wijzigen en verwijderen van tupels/instances. Tevens wordt een menustructuur gegenereerd om deze onderhoudsfuncties te kunnen gebruiken.

Navigatie door de database

Er zijn applicatiegeneratoren die op basis van de gegevensstructuur een compleet navigatiemechanisme genereren. Voor het verkrijgen van deze functionaliteit hoeft dus niets te worden geprogrammeerd. Als gebruiker van het systeem kan men vanuit elke plaats in de database met behulp van een functietoets of een ‘push-button’ een menu op het scherm krijgen waarin de ‘belendende’ tabellen staan opgesomd. Belendende tabellen zijn de parents en children van de actuele tabel. Door een menu-optie te kiezen kan men navigeren. In sommige gevallen, bijvoorbeeld bij de parent-tabel, is de menu-optie vervangen door een directe look-up-functie met behulp van een functietoets of een push-button.

Figuur 21 illustreert welke navigatiepaden men krijgt wanneer men vanuit een medewerker op een bepaalde functietoets drukt. Elk van de navigatiepaden is te selecteren via een menu-optie. Kiest men een bepaald pad, bijvoorbeeld de afdeling waarvoor de medewerker werkt, dan krijgt men vervolgens die afdeling te zien, van waaruit dan weer de navigatiepaden voor afdelingen kunnen worden geselecteerd.



Figuur 21 Navigatiepaden

Als een generator de beide bovengenoemde functionaliteiten (gebruik tabellen en navigatie door de database) kan genereren, is er dus sprake van een werkende en ook bruikbare applicatie. De gehele database is benaderbaar en bruikbaar. Een dergelijk systeem, gebaseerd op een goed gegevensmodel of objectmodel, kan dienen als prototype (eerste systeemversie), met behulp waarvan de overige functionaliteit kan worden gespecificeerd en toegevoegd.

Overige functionaliteit

De overige functionaliteit wordt ook, maar dan expliciet, in het gegevensmodel of objectmodel gemodelleerd. Afhankelijk van de kwaliteiten van de gebruikte applicatie-generator hoeft men hier meer of minder inspanning voor te verrichten. Het komt neer op parametriseren, declareren of programmeren.

Veel functionaliteit ligt besloten in zogenaamde beperkingregels (constraints). Hoe meer beperkingregels er gelden (en gemodelleerd worden), des te specifiek wordt het model. Deze regels worden zoveel mogelijk als informatieregels aan het informatiemodel toegevoegd. Voorbeelden van deze regels zijn identificerende attributen, verplichte attributen, minimum- en maximum cardinaliteiten van associaties, waardebeperkingen en transitieregels.

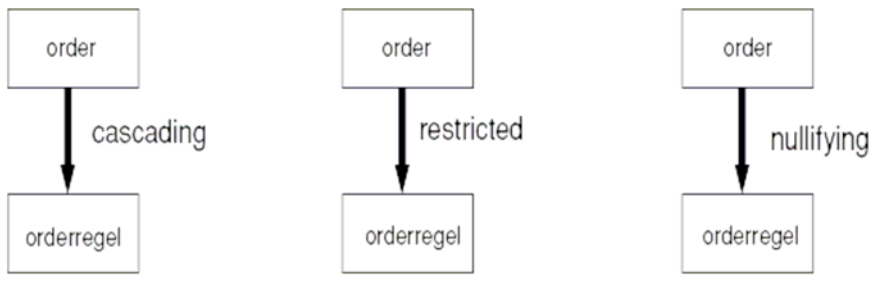
‘Update & delete’-regels

Behalve de hiervoor genoemde beperkingsregels zijn er ook regels waarmee het dynamisch gedrag van de gegevensstructuur wordt gemodelleerd. Bijvoorbeeld: wat moet er gebeuren met orderregels als de bijbehorende order wordt verwijderd? Deze regels worden doorgaans de ‘update & delete’-regels genoemd en worden als declaraties aan het gegevensmodel toegevoegd. Een associatie is ‘restricted’, ‘cascading’ of ‘nullifying’ ten aanzien van het ‘update’- of ‘delete’-gedrag. Deze begrippen betekenen het volgende:

- cascading: wanneer een tupel van een ‘parent’-tabel uit de database wordt verwijderd, worden alle gerelateerde tupels van de ‘child’-tabel ook verwijderd, en elke wijziging van de primaire sleutel van een tupel uit de ‘parent’-tabel resulteert in een wijziging van de vreemde sleutel van alle gerelateerde tupels van de ‘child’-tabel;
- restricted: verwijdering van een tupel van een ‘parent’-tabel is niet toegestaan zolang er gerelateerde tupels van de ‘child’-tabel bestaan, en wijziging van de primaire sleutel van een tupel van de ‘parent’-tabel is niet toegestaan zolang er gerelateerde tupels van de ‘child’-tabel bestaan;
- nullifying: wanneer een tupel van een ‘parent’-tabel uit de database wordt verwijderd, blijven gerelateerde tupels van de ‘child’-tabel in de database bestaan zonder een referentie naar een tupel van de ‘parent’-tabel, en een wijziging van de primaire sleutel van een tupel uit de ‘parent’-tabel heeft tot gevolg dat alle gerelateerde tupels uit de ‘child’-tabel zonder referentie blijven staan (de vreemde sleutel krijgt de waarde NULL).

Figuur 22 toont de volgende situaties met betrekking tot het ‘delete’-gedrag:

- cascading: wanneer een order wordt verwijderd, worden alle gerelateerde orderregels ook verwijderd;
- restricted: het verwijderen van een order is niet toegestaan zolang er nog aan deze order gerelateerde orderregels voorkomen;
- nullifying: wanneer een order wordt verwijderd, blijven de aan deze order gerelateerde orderregels bestaan, echter zonder referentie naar een order.



Figuur 22 'Update & delete'-regels

Afleidingsregels

Een apart te beschouwen categorie regels wordt gevormd door de zogenaamde afleidingsregels. Bijvoorbeeld: als A geldt, dan moet ook B gelden (implicatie) en ook omgekeerd (equivalentie); of: het orderbedrag is de som van de bedragen van de bijbehorende orderregels. Een dergelijk regel in het informatiemodel beweert alleen maar dat iets in de werkelijkheid zo is. Bij de systeemspecificatie wordt (in overleg met de gebruiker) bepaald hoe de regel wordt geïmplementeerd. Eerst wordt bepaald of het afleidbare gegeven wel of niet in de database wordt opgenomen. Vervolgens wordt bepaald of het afleidbare gegeven wordt afgeleid en wanneer deze afleiding wordt uitgevoerd, of dat de regel wordt gebruikt voor controle van het gegeven als de gebruiker het expliciet inbrengt. In het laatste geval is het weer een beperkingsregel. De voornaamste overweging hierbij is de vraag of het betreffende gegeven geoptimaliseerd moet zijn voor 'updating' of voor 'retrieval'.

Internal events

De laatste categorie functionaliteit wordt hier aangeduid met de term 'internal events': als er iets gebeurt in het systeem, moeten er een of meer acties plaatsvinden. Bijvoorbeeld: de actuele voorraad komt onder het bestelniveau en er moet automatisch een bestelling worden gecreëerd. Ook dit soort functionaliteit wordt op het gegevensmodel aangebracht, al zal dat meer op programmeren dan parametriseren lijken. Maar het mechanisme, met behulp van zogenaamde triggers, is bij alle regels hetzelfde.

Event-getriggerde regels

Bij het specificeren van regels leggen we vast:

- welke actie(s) moet(en) worden uitgevoerd;
- welk voorval ('event') aanleiding is tot uitvoering;
- welke voorwaarden gelden voor uitvoering.

Een op die wijze gespecificeerde regel noemen we een 'event-getriggerde regel'. We kunnen er een algemeen geldige vorm aan geven:

```

ON    <voorval>
IF    <conditie>
THEN  <actie>
  
```

Het voorval activeert de regel. Een voorval kan onder andere zijn:

- het kiezen van een menu-optie of een functietoets;
- juist vóór (PRE) of na (POST) een UPDATE, DELETE, INSERT van een tupel;
- juist vóór of na een CHANGE van een attribuut.

De conditie moet gelden om de regel te kunnen uitvoeren. Voorbeelden van condities zijn:

- statusvalidatie, bijvoorbeeld: de order moet geaccepteerd zijn;
- bestaan van een associatie (vreemde sleutel 'NOT NULL');
- een expressie, bijvoorbeeld: $A > B$.

De actie is het procesgedeelte van de regel. Enkele voorbeelden zijn:

- het controleren van een waarde;
- het berekenen of afleiden van een waarde;
- het uitvoeren van een procedure.

Voorbeeld:

```
ON    POST-INSERT van orderregel  
IF    aantal van orderregel <= voorraad van artikel  
THEN  voorraad van artikel := voorraad van artikel – aantal van order/regel
```

```
ON    POST-CHANGE van voorraad van artikel  
IF    voorraad van artikel < bestelniveau  
THEN  procedure Plaats-bestelling
```

Complexe regels

Complexe regels zijn regels die meer entiteitstypen of objecttypen raken, bijvoorbeeld een berekening waarbij een groot gedeelte van het gegevensmodel/objectmodel betrokken is. In het algemeen zijn er twee oplossingsrichtingen voor dergelijke regels:

- implementeer de regel als één proces;
- distribueer de regel als kleine regels over het gegevensmodel/objectmodel.

De implementatie als één proces bestaat uit een programma dat wordt geactiveerd door een voorval, bijvoorbeeld een functietoets of menu-optie. Het resultaat van het uitvoeren van de regel kan worden gekoppeld aan een virtueel veld of aan een databaseveld. Een mogelijk nadeel van deze oplossing is een relatief lange berekeningstijd.

De implementatie van de gedistribueerde oplossingsrichting bestaat uit twee delen:

- distribueer het rekenproces als informatieregels over het gegevensmodel;
- bepaal de voorvallen (events) die de informatieregels activeren.

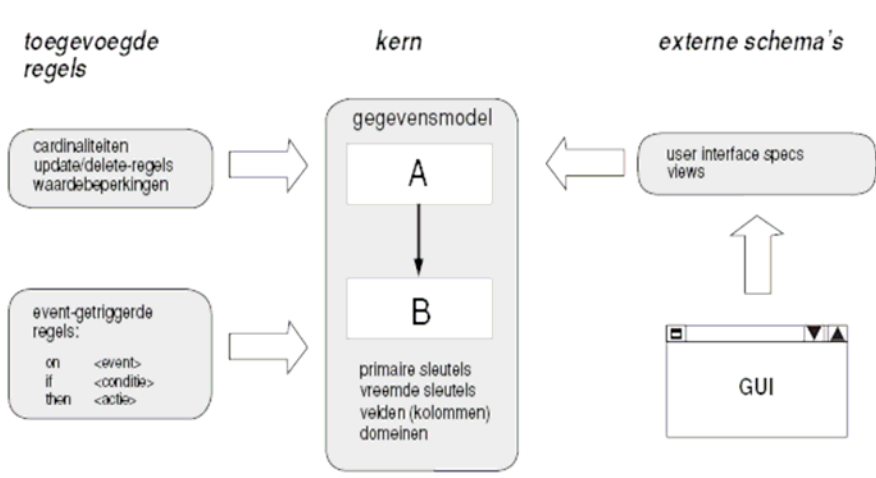
De voortschrijdende database-technologie, onder andere het ontstaan van object-georiënteerde DBMS'en, bevordert de distributie van regels over het gegevensmodel. Sommige DBMS'en, waaronder Oracle, ondersteunen al 'stored procedures', wat betekent

dat het mogelijk is om regels te koppelen aan objecten in de database. Deze regels worden dan bewaakt door het DBMS, onafhankelijk van de processen die de objecten benaderen. Ook kan de computertijd die de uitvoering van een regel vergt worden gedistribueerd door verschillende voorvallen te definiëren op delen van de regel. Verder kunnen andere processen de resultaten van deze gedistribueerde delen gebruiken.

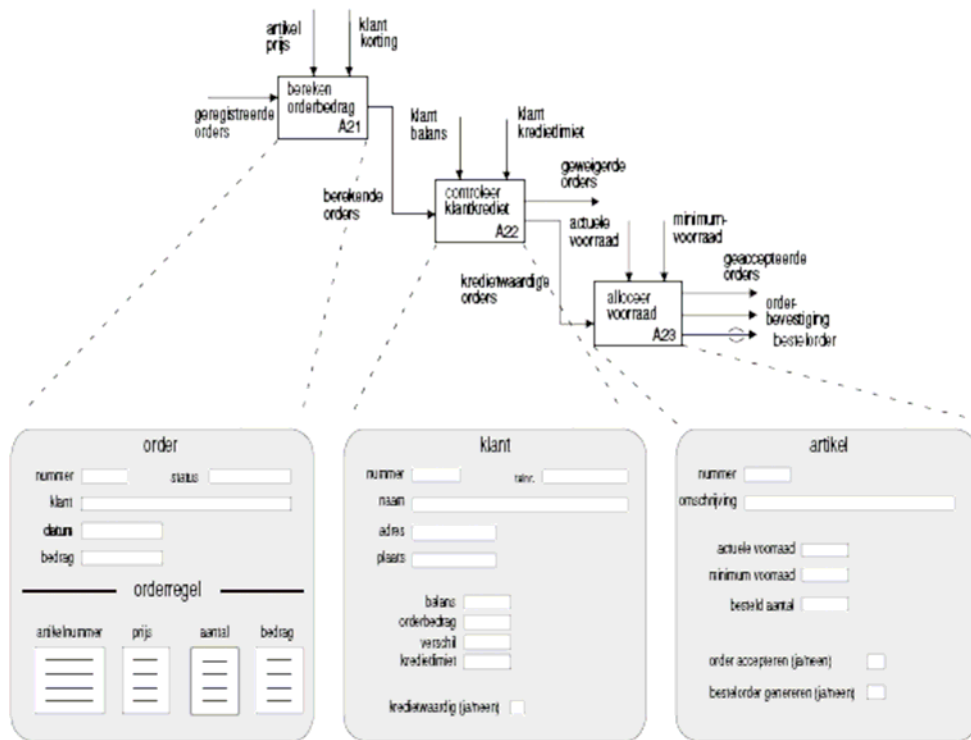
UML versie 2.0 is ten behoeve van Model Driven Architecture verrijkt om een model executeerbaar te maken. Het achterliggende idee is om, in plaats van het model te transformeren naar een implementatiemodel, het model direct te kunnen executeren op het doelplatform. Om modellen te kunnen executeren is een formele specificatietaal nodig om het gedrag van het systeem te beschrijven. In feite dus een taal voor het vastleggen van de informatieregels. Aan de hand van deze beschrijving en de modellen kan vervolgens het complete systeem worden gegenereerd naar bijvoorbeeld C++, Java of SQL. Voorbeelden van een formele specificatietaal zijn Action Specification Language, ontwikkeld door Kennedy Carter, en Object Constraint Language, gedefinieerd door Klasse. Executeerbaar UML maakt een vroegtijdige validatie van modellen mogelijk aan de hand van werkende systeemversies. De belofte van executeerbaar UML is fascinerend en onderstreept de eerdere conclusie dat bij de toepassing van applicatiegeneratoren de focus van softwareontwikkeling verschuift naar het modelleren van de bedrijfswerkelijkheid en de gewenste systeemfunctionaliteit.

In het algemeen moet een keuze worden gemaakt hoe een complexe regel wordt geïmplementeerd. Echter, hoe meer men in staat is complexe regels te distribueren, des te robuuster en flexibeler zal het informatiesysteem zijn. Een voorbeeld van een complexe regel is uitgewerkt in appendix B.

Het uiteindelijke resultaat – een gegevensmodel met daarop afgebeelde regels of een objectmodel met ingesloten functionaliteit – is te beschouwen als een executeerbaar model (zie figuur 23).



Figuur 24 Het specificatiemodel



Figuur 25 Afleiden van views aan de hand van het bedrijfsactiviteitenmodel

2.7 Samenvatting

In dit hoofdstuk zijn de bij een MAD-aanpak toegepaste modeltypen en hun onderlinge samenhang beschreven. Uitgaande van een activiteitenmodel worden entiteitstypen/business-objecttypen en hun associaties en attribuuttypen gemodelleerd in een informatiemodel/business-domeinmodel. Van dit informatiemodel/business-domeinmodel wordt een relationeel gegevensmodel/objectmodel afgeleid dat de basis vormt voor applicatiegeneratie. Functionaliteit wordt in de vorm van regels aan het gegevensmodel/objectmodel toegevoegd (constraints en eventgetriggerde regels).

Tevens is in dit hoofdstuk de structuur van het specificatiemodel beschreven. Deze structuur is onafhankelijk van de ontwikkelomgeving (lees: applicatiegenerator) die wordt ingezet. Uit praktische ervaringen blijkt dat specificatiemodellen volgens deze structuur op verschillende applicatiegeneratoren kunnen worden afgebeeld. Het specificatiemodel bestaat uit:

- de kern: het relationele gegevensmodel/objectmodel;
- toegevoegde regels;
- externe schema's.

Dit specificatiemodel vormt voor de ontwikkelaar de relevante 'source code'. Alle ontwikkeling en onderhoud vinden plaats op dat model. De ontwikkelaar kan het systeem terecht beschouwen als een executeerbaar specificatiemodel. Onze ervaringen leren dat dit een tamelijk ingrijpende 'mind shift' vergt van (met name ervaren) software-engineers. Zij moeten leren informatiesystemen te beschouwen als een werkende afbeelding (model) van een werkelijkheid in plaats van een op zichzelf staand hulpmiddel dat door procedurele en conceptuele transformaties van gebruikerswensen in uitvoerbare code is omgezet.

Omdat de structuur van de gegenereerde applicatie gevormd wordt door het gegevensmodel/objectmodel, dat op zijn beurt weer gelijkvormig is met het (bedrijfs) informatiemodel, wordt het onderhoud op de applicatie veel efficiënter. Immers, een verandering in de werkelijkheid wijst ondubbelzinnig naar de overeenkomstige plaats in het systeem.

3 De MAD-projectaanpak

In dit hoofdstuk wordt de MAD-projectaanpak nader toegelicht. Omwille van de leesbaarheid van dit hoofdstuk wordt met een informatiemodel ook een businessdomein-model bedoeld en wordt met een gegevensmodel ook een objectmodel (klassendiagram) bedoeld. Eén en ander overeenkomstig de terminologie zoals weergegeven in de tabel op pagina 11.

3.1 Voorafgaand aan een MAD-project

Informatieplanning

Projecten zijn meestal gebaseerd op een informatieplan. Een informatieplan omvat:

- de afbakening van het bedrijfsproces dat wordt ondersteund door de toekomstige systemen;
- een bedrijfsactiviteitenmodel;
- blauwdrukken van het informatiemodel;
- een architectuur voor de informatiesystemen;
- een architectuur voor de technische infrastructuur;
- een automatiseringsplan.

Wanneer een dergelijk plan niet voorhanden is, is het alsnog uitvoeren van informatieplanning wenselijk voor het project. Informatieplanning wordt dan uitgevoerd in een relatief korte tijd (3-4 maanden) en voor een specifiek bedrijfsonderdeel. Deze organisatorische eenheid mag niet te groot zijn, bijvoorbeeld een afdeling of een fabriek. Het informatieplan wordt opgesteld door een kleine groep personen bestaande uit stafleden en automatiseringspersoneel.

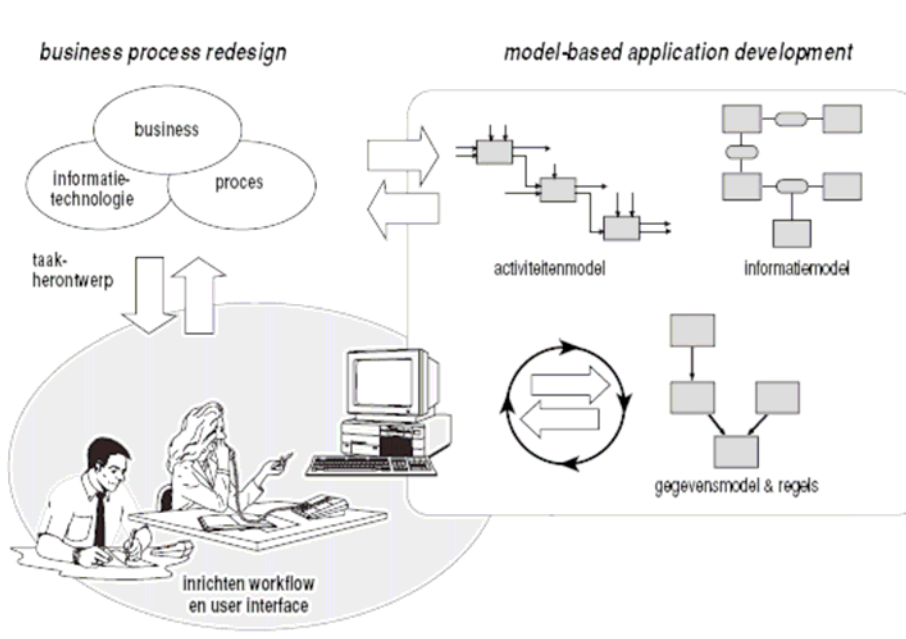
Business Process Redesign (BPR)

Business Process Redesign of Business Process Re-engineering (BPR) is het opnieuw inrichten van het bedrijfsproces, waarbij informatietechnologie al bij het ontwerp van het bedrijfsproces wordt beschouwd. De technische mogelijkheden staan aan de basis van het herontwerp, waarbij nieuwe technologieën worden toegepast in het nieuwe bedrijfsproces.

Tijdens een BPR-traject worden drie aspecten spiraalsgewijs op elkaar afgestemd: Business (heroverwegen van bedrijfsstrategie en product/markt-combinaties), Proces (herontwerp) en Informatietechnologie (nieuwe toepassingen). Een BPRtraject kan al veel bruikbare input leveren voor een MAD-project, vooral de beschrijving van het nieuwe proces, de workflow en de daarbinnen gedefinieerde taken van de gebruikers. Zeker wanneer de procesbeschrijvingen in de vorm van procesmodellen zijn opgesteld, verschaft een BPR-traject een vliegende start voor een MAD-project.

Figuur 26 illustreert het verband tussen BPR en MAD. Dit verband is tweeledig:

- **Bedrijfsproces en informatiebehoefte**
Tijdens een BPR-traject wordt het nieuwe bedrijfsproces beschreven. Bovendien wordt de informatiebehoefte in termen van objectklassen of entiteitstypen vastgelegd. Een MAD-project start doorgaans met het modelleren van het bedrijfsproces vanuit de invalshoek van informatie-uitwisseling. Het modelleren van bedrijfsactiviteiten kan worden gezien als een verdere detaillering van het procesmodel uit het BPR-traject.
- **Workflow en taken**
Tijdens een BPR-traject worden de workflow en de daarbinnen gedefinieerde taken vastgelegd. Deze beschrijving kan als uitgangspunt dienen bij het ontwerpen van de user interface (externe schema's).



Figuur 26 Integratie van BPR en MAD

Het BPR-traject hoeft niet noodzakelijkerwijs te zijn afgerond voordat een MAD-project wordt gestart. Integendeel: er ontstaan juist synergetische effecten wanneer beide geïntegreerd en deels parallel worden uitgevoerd. Gegeneerde systeemversies kunnen dan worden gebruikt om het nieuwe bedrijfsproces te visualiseren. Het iteratieve karakter van MAD sluit aan bij de spiraalsgewijze beschouwing van business, processen en informatietechnologie. Bovendien maakt MAD een snelle invoering van het bedrijfsproces mogelijk door de hoge productiviteit en, daarmee samenhangend, de gunstige 'time-to-market' van de applicaties.

Architectuur

- Het belang van architectuurgedreven softwareontwikkeling is toegenomen. Binnen een organisatie zijn drie niveaus van architectuur te onderscheiden:
- de business-architectuur – het geheel aan primaire, secundaire (ondersteunende) en bestuurlijke bedrijfsprocessen;
- de applicatiearchitectuur – het geheel aan softwareapplicaties ter ondersteuning en uitvoering van bedrijfsprocessen en hun onderlinge samenhang;
- de technische infrastructuur – het geheel aan hardware, netwerken, besturingssystemen, middleware, DBMS en ontwikkelomgevingen en hun onderlinge samenhang.

MAD is vooral gericht op de realisatie van de applicatiearchitectuur. Veelal zal voor de start van een project de business-architectuur in kaart zijn gebracht en zullen blauwdrukken van het informatiemodel voorhanden zijn.

3.2 Projectplanning en -beheersing

In deze paragraaf worden enkele beheersaspecten van MAD-projecten belicht zonder de pretentie van volledigheid. Voor de aspecten van projectmanagement en kwaliteitsbeheersing wordt verwezen naar het Handboek Projectmanagement [Bosschers].

Als geheugensteuntje is het acroniem MOSQUITO bedacht. Bij iedere stap van de beheerscyclus (voorbereiden, uitvoeren, toetsen en beslissen) kan de projectleider dit acroniem doorlopen om te controleren of hij aan alle beheersaspecten heeft gedacht:

- money – geld
- safety – risico's
- quality – kwaliteit
- information – informatie
- time & resources – tijd en hulpbronnen
- organization – organisatie en mensen

Money (geld)

In een MAD-project, in het bijzonder wanneer prototyping wordt toegepast om de specificaties helder te krijgen, is het nauwelijks mogelijk om functiepuntenanalyse (FPA) in haar huidige vorm als schattingsmethode toe te passen. FPA is immers gebaseerd op gedetailleerde specificaties van systeemfuncties. Echter, bij een MAD-aanpak ontbreken deze. Desondanks is al in een vroeg stadium inzicht in de omvang van het systeem gewenst en moet vaak in dit stadium al een enigszins betrouwbare schatting worden afgegeven van de benodigde capaciteit. Er moet dan dus worden begroot op basis van het informatiemodel of, later, op basis van het wat meer gedetailleerde gegevensmodel.

Om toch al in een vroeg stadium, bijvoorbeeld na de bedrijfsanalyse, een kwantificering te kunnen maken van de omvang van het systeem, is bij de tot nu toe uitgevoerde MAD-projecten een alternatieve ‘telling’ toegepast. Deze wijze van tellen is niet theoretisch onderbouwd, maar is louter een lapmiddel bij gebrek aan beter. De telling wordt uitgevoerd op een informatiemodel en een globale beschrijving van informatieregels per entiteitstype. De beschrijving van de informatieregel hoeft nog niet compleet of gedetailleerd te zijn, bijvoorbeeld ‘berekening fabriekscapaciteit’ volstaat, zonder al precies te weten hoe dit moet worden berekend. Inzicht in de mate van complexiteit van de informatieregel is wel gewenst.

De telling gaat uit van een invoer-, uitvoer-, opvraag- en bestandsfunctie per entiteitstype. Aan de hand van het aantal informatieregels en de complexiteit hiervan wordt per entiteitstype en per functie een inschatting gemaakt van de classificering (eenvoudig, gemiddeld of complex). Vervolgens wordt deze classificering gekoppeld aan de bekende normering volgens FPA (laag, gemiddeld en hoog). Dit resulteert in een hoeveelheid functiepunten per entiteitstype, variërend van 17 tot 34.

Aanvullend worden de koppelingsfuncties geclassificeerd en genormeerd.

Ervaringen laten zien dat deze schattingen minder dan 10% afwijken van tellingen die achteraf op het gerealiseerde systeem zijn uitgevoerd. De systemen die op deze alternatieve wijze zijn geteld varieerden in omvang van 300 tot 2000 functiepunten.

Daarnaast dient in een contractuele relatie tussen de opstellers van de specificaties en de bouwers van het systeem aandacht te worden besteed aan het verschijnsel dat ‘complete, allesomvattende’ specificaties (mijlpaaldocument, dat als contractbasis kan dienen), ontbreken bij een MAD-project. Om een en ander toch te beheersen moet op voorhand worden afgesproken welke functionaliteit wordt opgeleverd. Hierbij moet onderscheid worden gemaakt tussen eenvoudige functionaliteit die wordt gegenereerd en de overige functionaliteit, zoals afleidingsregels, rapporten en interfaces naar andere systemen. Uitgaande van een redelijk stabiel informatiemodel zal het weinig moeite kosten om de eerste categorie van functionaliteit op te leveren. Voor het realiseren van de tweede categorie zal men meer tijd en moeite kwijt zijn. Als beheersinstrument zou men van tevoren het aantal iteraties of een bepaald budget kunnen afspreken. Voor het goed hanteren van dit laatste instrument moet er sprake zijn van een vertrouwensrelatie tussen opdrachtgever en opdrachtnemer.

Bij het schatten van capaciteit is het aan te bevelen om gedifferentieerde productiviteitsratio’s te hanteren. De eerste categorie – triviale functionaliteit en eenvoudige beperkingsregels – wordt gerealiseerd met een hoge productiviteit (bijvoorbeeld 20 functiepunten per uur). De tweede categorie wordt gerealiseerd met een ‘normale’ productiviteit voor een 4GL-programmertaal of een objectgeoriënteerde taal als Java of C#.

De beperkte toepassing van FPA als schattingsmethode – met name bij een objectgeoriënteerde aanpak – wordt voor een deel opgevangen door cosmic full function points. Deze methode biedt een paar voordelen boven FPA, onder andere de eenvoud van meten en een betere behandeling van de complexiteit van software. Helaas is deze

maateenheid niet vergelijkbaar met het ‘klassieke’ functiepunt. Dit impliceert dat ervaringscijfers (productiviteitsratio) opnieuw moeten worden opgebouwd.

Safety (risico's)

Naast de ‘normale’ risico's kunnen mogelijk de volgende risico's een rol spelen:

- Het projectteam beschikt over onvoldoende kennis en ervaring in modelleren;
- Het projectteam onderschat het modelleren en zet te vroeg een generator in;
- De voor de generator benodigde hardware en software wordt door de leverancier te optimistisch gesteld;
- Het projectteam denkt voldoende te hebben aan de manuals en gaat zonder adequate training met de generator aan de slag.

Wanneer dergelijke risico's van toepassing zijn, dienen maatregelen te worden genomen ter reductie van het risico. Te denken valt aan:

- initiële training en intensieve begeleiding;
- support en commitment van de leverancier.

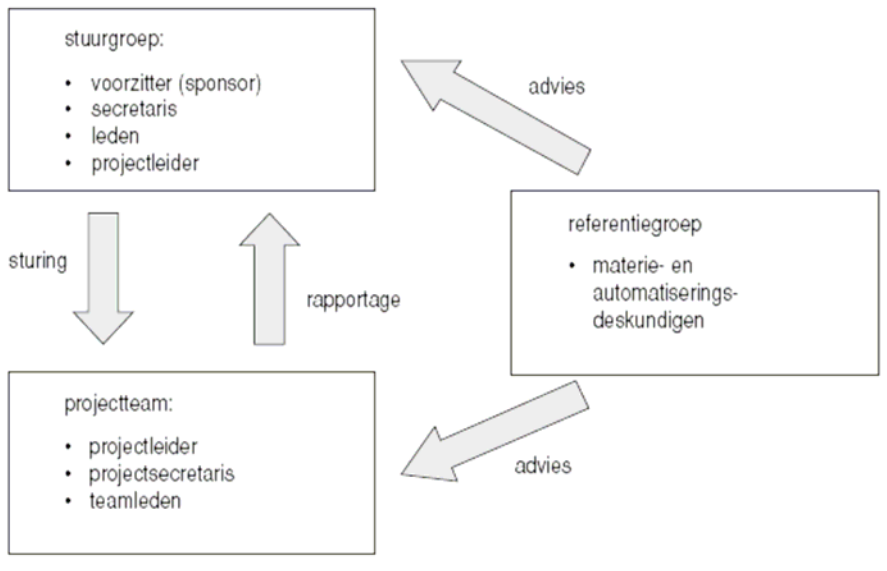
Daarnaast bestaat het gevaar dat gebruikers en ontwikkelaars tijdens hun afstembijeenkomsten extra functionaliteit definiëren ten opzichte van de oorspronkelijke afspraken ('creeping functionality'). Opdrachtgever en opdrachtnemer moeten zich terdege bewust zijn van dit gevaar en gezamenlijk afspraken maken over het instrumentarium voor projectmanagement als maatregel ter voorkoming van dit risico (zie bijvoorbeeld onder Time (tijd)).

Bij toepassing van een MAD-aanpak ontstaan reeds in een vroeg stadium, soms al tijdens het offertetraject, werkende versies van het te ontwikkelen systeem. De opdrachtgever kan hierdoor de indruk krijgen dat het systeem zo goed als klaar is, terwijl er 'slechts' sprake is van een eerste gegeneerde versie die voorziet in de triviale functionaliteit. Hierdoor kan bij de opdrachtgever een verkeerd beeld ontstaan over de productiviteit en de hoeveelheid werk die nog moet worden verricht om een installeerbaar systeem op te leveren.

Opdrachtgever en ontwikkelaars kunnen verschillende beelden hebben van het moment van oplevering. De iteratieve werkwijze nodigt uit tot het steeds verder aanpassen van het ontwikkelde model en het daaruit gegeneerde systeem. Leverancier en opdrachtgever moeten daarom goede afspraken maken over het moment van oplevering van het systeem en van de wijze van decharge van de opdrachtnemer.

Quality (kwaliteit)

Het uitvoeren van kwaliteitsaudits op de modellen komt de kwaliteit van de specificaties ten goede. Hiervoor kan een referentiegroep worden ingesteld, die het projectteam bijstaat en de stuurgroep adviseert over de inhoudelijke kwaliteit van de specificaties. Hiertoe worden regelmatig audits of inspecties gehouden. De referentiegroep wordt bemand door materie- en automatiseringsdeskundigen. Onder de laatsten moeten bij voorkeur enkele professionals op het gebied van modelleren zitten. Figuur 27 illustreert de plaats en de rol van de referentiegroep in de projectorganisatie.



Figuur 27 Projectorganisatie

Information (informatie)

Het productdossier, zijnde alle programmatuur en documentatie die onderdeel zijn van het product, is bij een MAD-project aanzienlijk minder omvangrijk dan bij een project dat volgens het klassieke waterval-scenario wordt uitgevoerd en waarbij het gebruikelijk is dat ‘alle’ specificaties gedetailleerd op papier worden gezet.

Time (tijd)

De incrementele ontwikkelbenadering geeft goede handvatten voor een tijdplanning. De planning gaat ervan uit dat op zekere momenten bepaalde resultaten worden opgeleverd. Bij MAD zijn dit de elkaar opvolgende incrementen van het in ontwikkeling zijnde informatiesysteem.

Een voorbeeld van een beheersmethode is ‘Time Box Management’ [Martin]. Time Box Management is een eenvoudige, maar doeltreffende methode voor projectmanagement. Het is niets anders dan het bepalen van onwrikbare mijlpalen: rotsvaste, onbeweegbare en niet-onderhandelbare tijdstippen waarop iets gereed moet zijn. Time Box Management gaat ervan uit dat de belangrijkste componenten van het informatiesysteem het eerst worden ontwikkeld. Komt men onverhoopt toch in de knel bij de oplevering van de resultaten, dan zijn in ieder geval de meest relevante delen beschikbaar binnen de bepaalde time box. In combinatie met de incrementele benadering kan het restant in volgende incrementen (in volgende time boxes) worden ontwikkeld. Bij de planning van het project kan men het aantal time boxes of iteraties op voorhand vaststellen.

Time boxing wordt vaak toegepast in combinatie met het toekennen van prioriteiten aan functionaliteit. Een voorbeeld is het MoSCoW-principe, waarbij de gewenste functionaliteit wordt verdeeld in vier categorieën: Must have, Should have, Could have en Won’t have. Minder bekend, maar net zo doeltreffend om de franjefunctionaliteit (nice-to-haves) te

beperken, is ‘parkeren en schrappen’. Hierbij wordt een direct verband gelegd tussen de te ontwikkelen functionaliteit en de waarde ervan voor de organisatie, met als doel de gebruiker een eigen budgettaire verantwoordelijkheid te geven.

Resources (hulpbronnen)

De kern van de ontwikkelomgeving wordt gevormd door de applicatiegenerator. Dat er ook gebruik wordt gemaakt van zogeheten ‘upper-CASE tools’ is niet zo vanzelfsprekend als het op voorhand lijkt. Immers, het uiteindelijke specificatiemodel wordt opgeslagen en onderhouden in de repository van de applicatiegenerator. Het gebruik van ‘upper-CASE tools’ is meer gericht op het vastleggen van de analysemodellen. Zij kunnen als zodanig worden beschouwd als handige, ‘intelligente’ tekenhulpmiddelen. In paragraaf 4.3 gaan we hier nader op in.

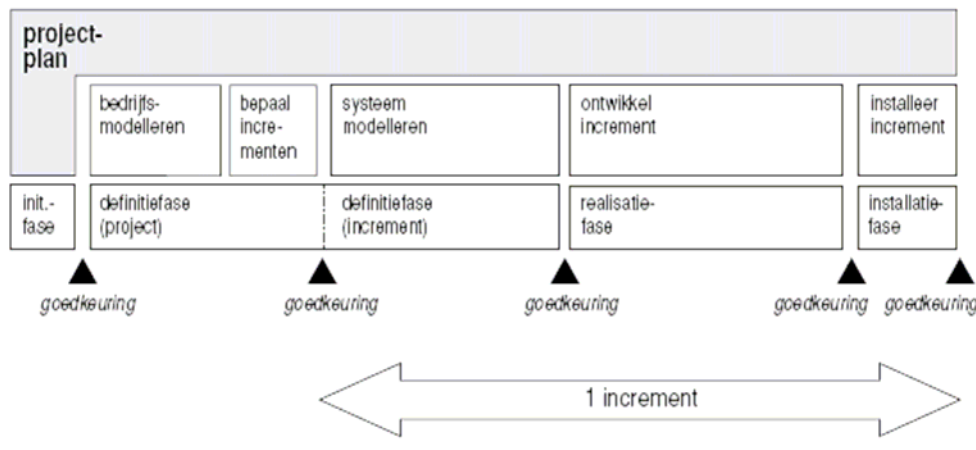
Organization (organisatie en mensen)

MAD-projecten worden bij voorkeur uitgevoerd door kleine projectteams. Natuurlijk is de bemanning mede afhankelijk van de omvang en complexiteit van de applicatie, maar over het algemeen is zij breed inzetbaar, omdat de ontwikkelactiviteiten elkaar snel opvolgen en zich een aantal keren herhalen en voortdurend met elkaar in wisselwerking zijn. Het projectteam is in staat om op het conceptuele niveau van bedrijfs- en informatiemodellen te werken en een volwaardige gesprekspartner te zijn voor de gebruikersorganisatie. Dit impliceert dat het beschikt over analytische vaardigheden en in staat is om aangedragen materiekennis te vertalen naar (informatie)modellen. Daarnaast beschikt het projectteam over voldoende technische kennis om een volwaardige applicatie te modelleren en te genereren. Dit betekent dat het met de applicatiegenerator kan omgaan en kennis moet hebben van de onderliggende techniek (Java, J2EE, Microsoft .NET, SQL, DBMS). Specialisten maken geen deel uit van het projectteam, maar zijn op afroep beschikbaar. Het is hun taak om het projectteam te ondersteunen met hun specialistische kennis, bijvoorbeeld database tuning, interface-protocol, GUI/ergonomie, enzovoort.

Martin introduceerde de Swat-teams (Skilled With Advanced Tools) – teams bestaande uit twee tot vijf allround software engineers die van het begin tot het eind bij het project betrokken zijn. Het idee is verder gecultiveerd in Atos Origins Software Development & Maintenance Center – een hedendaagse variant van een softwarefabriek [Kranenburg].

3.3 Projectinrichting

Figuur 28 toont een mogelijke inrichting en fasering voor een MAD-project. Het project is opgedeeld in een initiatiefase en drie werkfasen: definitie, realisatie en installatie. Voor elke fase worden de normale projectmanagementactiviteiten uitgevoerd. Deze blijven hier buiten beschouwing; hiervoor wordt verwezen naar het Handboek Projectmanagement [Bosschers].



Figuur 28 Projectinrichting

In de initiatiefase wordt het plan van aanpak gemaakt en wordt het project voorbereid. Het is aan te bevelen om een kick-off-meeting te organiseren, waarin de MAD-aanpak, de wijze van modelleren en specificeren en de te hanteren technieken worden toegelicht. Het is altijd goed om een dergelijke aftrap te organiseren voor de projectteamleden, maar het is ook nuttig voor participerende materiedeskundigen en gebruikersvertegenwoordigers.

De definitiefase is onderverdeeld in één fase voor het project en één fase voor het increment. In het geval dat er geen afzonderlijke incrementen zijn – het systeem wordt in zijn geheel gebouwd en opgeleverd – vallen deze fasen samen.

Tijdens de definitiefase voor het project worden de bedrijfs- en informatiemodellen opgesteld. Tijdens de definitiefase voor een increment worden de specificaties (systeemmodellen) van dat increment op een iteratieve wijze gedefinieerd. Met behulp van gegenereerde systeemversies worden deze specificaties gevalideerd.

Tijdens de realisatiefase wordt het increment/systeem verder uitgebreid met onder andere rapporten, interfaces met andere systemen, op taken afgestemde user interface en autorisatie.

In de installatiefase wordt het increment geïntegreerd met andere systeemdelen en vindt installatie in de gebruikersomgeving plaats gevolgd door de acceptatietest. Hierna kan worden overgegaan tot implementatie van het systeem (oplevering, opleiding, ingebruikneming en nazorg).

We stellen nadrukkelijk dat hier een mogelijke fasering is gegeven. Deze zal immers veelal afhangen van de organisatorische context waarbinnen het MAD-project wordt uitgevoerd, zoals scheiding opdrachtgever/opdrachtnemer, al dan niet 'ge-outsourced'. De scheiding tussen opdrachtgever en opdrachtnemer wordt meestal gelegd tussen de definitiefase en de realisatiefase. Bij het uitvoeren van een MAD-project is deze scheiding weinig zinvol. Een scheiding tussen de project-definitiefase en de increment-definitiefase, of tussen bedrijfsmodelleren enerzijds en systeemmodelleren en realisatie anderzijds, lijkt meer voor de hand

te liggen. Wij zien bedrijfsmodelleren, en dan in het bijzonder de informatiemodellen, als een ‘eigen’ activiteit van de opdrachtgever, met name gericht op het beheer van een informatie-infrastructuur.

Omdat de scheiding opdrachtgever/opdrachtnemer niet bij elk project hetzelfde is, leggen wij de nadruk op de ontwikkelactiviteiten die tijdens een MAD-project worden uitgevoerd. Deze worden in de volgende paragraaf beschreven.

3.4 Projectuitvoering

De volgende ontwikkelactiviteiten (1 tot en met 9) worden tijdens een MAD-project uitgevoerd.

1. Stel voor het applicatiegebied een bedrijfsactiviteitenmodel op

Gebruikersvertegenwoordigers (bij voorkeur stafleden en materiedeskundigen) dienen bij het modelleren van de bedrijfsactiviteiten betrokken te zijn. In het algemeen zullen zij niet getraind zijn in het lezen van bedrijfsactiviteitenmodellen. Wanneer er geen kick-off-meeting heeft plaatsgevonden, is het aan te bevelen om de doelstellingen van het modelleren van bedrijfsactiviteiten en de te hanteren tekentechniek te introduceren tijdens een eerste bijeenkomst tussen de informatieanalist en de gebruikersvertegenwoordigers. In de praktijk blijken bedrijfsactiviteitenmodellen een goed communicatiemiddel te zijn tussen de informatie-analist en de gebruikersvertegenwoordigers.

Het modelleren van bedrijfsactiviteiten is niet altijd noodzakelijk. Wanneer het bedrijfsproces bekend is (bijvoorbeeld bij een vervolgproject in dezelfde applicatie-omgeving) of bij de migratie van een bestaande applicatie, draagt het modelleren van bedrijfsactiviteiten niet veel bij. In een dergelijk geval kan men zich beter concentreren op het informatiemodel.

2. Stel een globaal informatiemodel op

Het globale informatiemodel beschrijft de entiteitstypen, hun attribuuttypen en hun identificerende attribuuttypen. Ofschoon in dit stadium nog niet elk attribuuttype en informatieregel bekend hoeft te zijn, moet de structuur wel juist zijn.

Ervaringen leren dat het moeilijk is om informatie- en gegevensmodellen met gebruikers af te stemmen. Het is daarom raadzaam om het informatiemodel af te stemmen door een gegenereerd prototype te laten zien. Het informatiemodel moet dan wel zo ver zijn uitgekristalliseerd dat dit prototype voor de gebruikers herkenbaar is. Tijdens een demonstratie van het prototype kunnen delen van het model simultaan worden getoond met het uit dat deel gegenereerde prototype. Op deze manier wordt het verband tussen het model en het systeem zichtbaar en duidelijk. Validatie van het informatiemodel op een dergelijke wijze zal eenvoudiger worden naarmate de gebruikers meer ervaring krijgen met deze benadering.

Het is verstandig om informatiemodellering altijd uit te voeren, zelfs in situaties waarbij een bestaand systeem wordt herbouwd ('re-development'). Ook al is in een dergelijk geval een gegevensstructuur aanwezig, bijvoorbeeld ontleend aan de 'oude' databasestructuur, dan is het toch verstandig om eerst het informatiemodel vanuit een bedrijfsgezichtspunt op te stellen, waarna het aanwezige gegevensmodel geverifieerd kan worden.

Bij bedrijfsmodelleren en systeemmodelleren wordt afstemming op bilaterale wijze vervangen door Joint Application Design (JAD-sessies). Door de systeemspecificaties te bespreken in groepsessies waarin alle relevante gebruikers(groepen) vertegenwoordigd zijn, kan snel overeenstemming over de specificaties worden bereikt. JAD-sessies moeten het heen-en-weer-afstemmen tussen informatieanalist enerzijds en meerdere individuele gebruikers anderzijds voorkomen.

3. Baken de grenzen van het systeem af

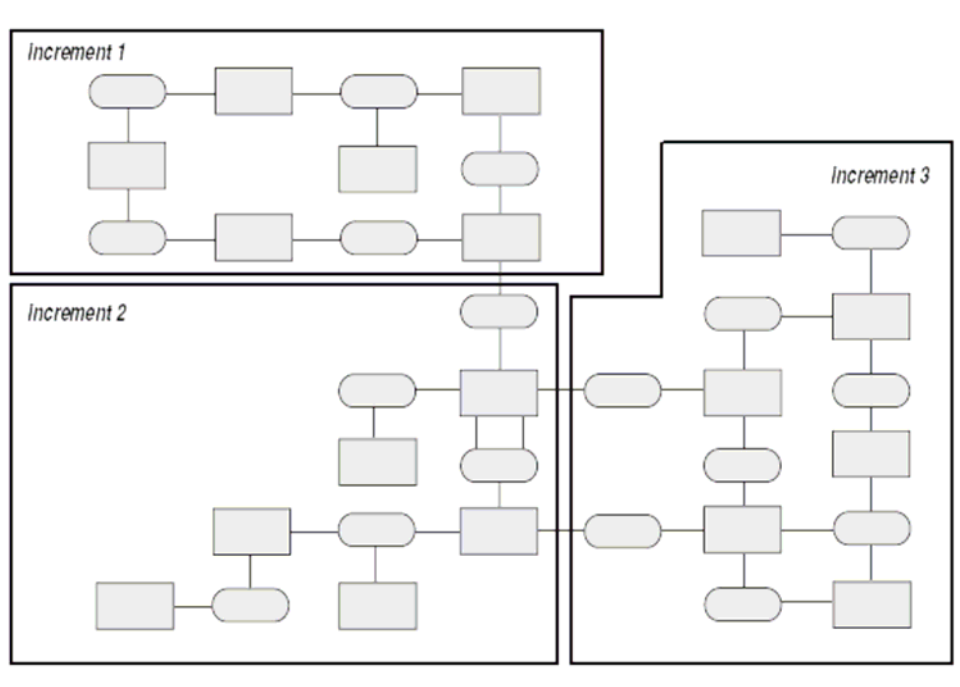
Bepaal binnen de grenzen van het applicatiegebied de omvang van het systeem door de bedrijfsactiviteiten te benoemen die worden ondersteund of uitgevoerd door het geautomatiseerde informatiesysteem.

Wijs de bedrijfsactiviteiten aan die communiceren met het geautomatiseerde informatiesysteem en bepaal de ingaande en uitgaande berichten. Bepaal de grenzen van het informatiemodel.

4. Definieer de incrementen

Incrementen worden herkend op basis van het informatiemodel. Figuur 29 illustreert dit. Een increment is bij voorkeur niet groter dan dat het in een halve manjaar kan worden gerealiseerd.

De volgorde waarin de incrementen verder worden gedetailleerd en ontwikkeld wordt bepaald op basis van afhankelijkheden tussen de incrementen en prioriteiten van de gebruikers. Doorgaans zal het eerste increment ten minste de 'kern' entiteitstypen bevatten van het betreffende applicatiegebied, bijvoorbeeld klanten, producten, leveranciers, werkstations en dergelijke. Opvolgende incrementen zullen, naast de kern-entiteitstypen, ook meer 'event-like' entiteitstypen bevatten, zoals klantorders, fabricage-orders, facturen, afhankelijk van de functionaliteit die het increment moet vervullen.

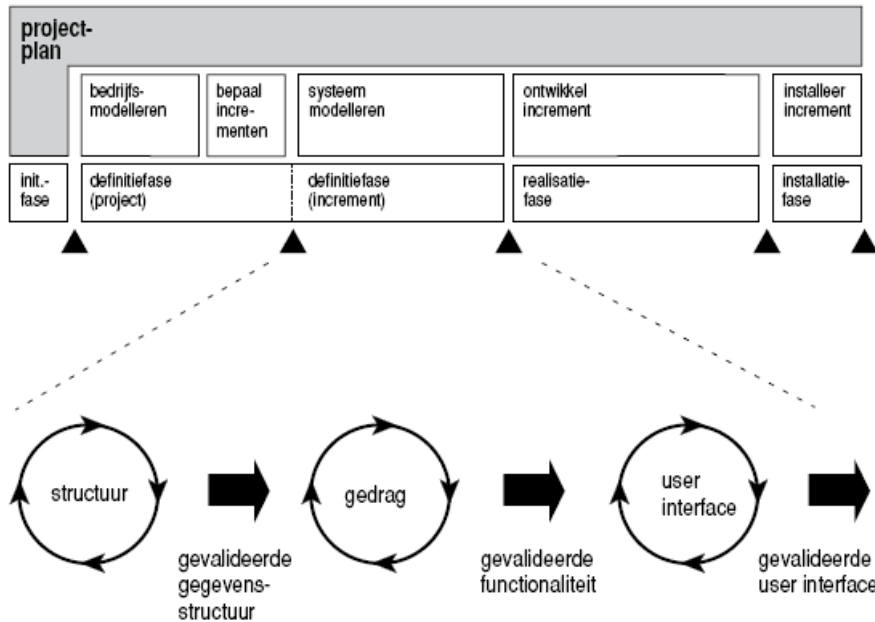


Figuur 29 Definitie van incrementen

Ervaring met de MAD-aanpak is een voorwaarde voor incrementele ontwikkeling. De gebruikersorganisatie moet bekend zijn met deze wijze van specificeren en ontwikkelen van informatiesystemen, opdat voor het realiseren van incrementen een prioriteitenvolgorde kan worden bepaald die strookt met de onderlinge afhankelijkheden.

Figuur 30 illustreert hoe de definitiefase voor één increment verloopt. Achtereenvolgens worden de aspecten structuur (gegevens), gedrag (functionaliteit) en user interface (externe schema's) iteratief gedefinieerd en gevalideerd. Per aspect kunnen hier een of meer iteraties voor benodigd zijn.

Voor elk increment worden de volgende ontwikkelactiviteiten (5 tot en met 9) uitgevoerd.



Figuur 30 Definitie van structuur, gedrag en user interface

5. Definieer de gegevensstructuur

Leid een relationeel gegevensmodel af uit het informatie(sub)model. Breng dit gegevensmodel in in de repository van de applicatiegenerator. Definieer en creëer zo nodig – afhankelijk van de generatiemogelijkheden van de applicatiegenerator – voor iedere tabel uit het gegevensmodel een basis-onderhoudsfunctie (functies voor het invoegen, opvragen, wijzigen en verwijderen van tupels voor iedere tabel) en navigatiefuncties (naar de belendende tabellen). Genereer een initieel systeem.

De herkenbaarheid van deze eerste systeemversie wordt vergroot door ‘foreign keys’ te voorzien van betekenisvolle attributen (gecontroleerde denormalisatie). Indien mogelijk kunnen eenvoudige informatieregels, zoals waardebeperkingen (domeinen), al worden ingebracht. Bovendien is het aan te bevelen om velden die als ‘non-displayable’ en ‘non-insertable’ zijn gedefinieerd, tijdelijk wel zichtbaar en muteerbaar te maken. Demonstreer deze eerste systeemversie aan de gebruikers en valideer de gegevensstructuur. Verfijn – waar nodig – het gegevensmodel. Herhaal de generatie/validatie-cyclus totdat een stabiele gegevensstructuur is ontstaan.

Bij de afstemming van het gegevensmodel worden de volgende zaken gecontroleerd:

- Wordt de informatiebehoefte afgedekt door het systeem (zijn alle relevante entiteitstypen opgenomen in het systeem)?
- Is de gegevensstructuur correct (bevat het systeem de juiste navigatiepaden)?
- Zijn de gekozen identificerende attributen juist?
- Zijn de kenmerkende attributen juist?
- Welke andere attributen moeten onderdeel van het systeem zijn?
- Welke attributen zijn verplicht?

6. Specificeer het gedrag

Het gedrag (de functionaliteit van het systeem) wordt gespecificeerd in de vorm van informatieregels. Bespreek de functionaliteit van het increment met de gebruikers, bij voorkeur tijdens groepsessies. Herhaal de generatie/validatiecyclus met de gebruikers totdat iedereen akkoord is met de gespecificeerde functionaliteit van het increment.

Ter bepaling van de informatieregels kan het volgende lijstje worden gehanteerd:

- Gelden er waardebeperkingen voor attributen, bijvoorbeeld minimum- en maximumwaarden?
- Zijn er onderlinge afhankelijkheden tussen associaties?
- Zijn er attributen waarvan de waarde kan worden afgeleid uit andere attributen?
- Welke statusovergangen zijn er toegestaan op attributen?
- Welke personen zijn geautoriseerd voor het toevoegen, opvragen, wijzigen en verwijderen van entiteiten en attributen?

Bovendien dient voor iedere associatie het ‘update’- en ‘delete’-gedrag te worden bepaald.

Eenvoudige informatieregels, zoals waardebeperkingen, worden doorgaans ‘meteen’ ingebracht. De andere informatieregels, zoals berekeningen en afhankelijkheden, worden beschreven en later tijdens de realisatiefase gebouwd.

7. Definieer de user interface

Bepaal de user interface (externe schema’s) aan de hand van de taken binnen de gedefinieerde workflow. Eenvoudige aanpassingen aan de gegenereerde user interface, bijvoorbeeld extra velden bij een ‘foreign key’ (gecontroleerde denormalisatie), kunnen in deze fase al worden gehonoreerd. Aanpassingen waarvoor wat meer komt kijken, bijvoorbeeld een spreadsheet-achtige interface of integratie met pakketten van derden, worden tijdens de realisatiefase gebouwd.

Kosmetische eisen – bijvoorbeeld de ‘look & feel’ van een GUI – worden bij voorkeur gehonoreerd door voorgedefinieerde instellingen (‘preferences’) te veranderen of sjablonen aan te passen. Het op grond van cosmetische wensen wijzigen van één extern schema moet, gelet op de onderhoudbaarheid, worden voorkomen.

Completeer de systeemspecificaties met de gewenste rapporten, eventuele batch jobs en de specificatie van interfaces met andere systemen. Stem deze specificaties met de gebruikers af, bij voorkeur in groepsessies.

8. Ontwikkel het increment

‘Ontwikkel’ betekent hier het uitbreiden van de laatst gegenereerde systeemversie met additionele functionaliteit. Vertaal de gespecificeerde informatieregels die nog niet in een voorgaande slag zijn meegenomen, in constraints en event-getriggerde regels. Test en (indien nodig) wijzig de geïmplementeerde constraints en event-getriggerde regels. Bouw of definieer en genereer (afhankelijk van de applicatiegenerator) de gewenste rapporten, user interface (externe schema’s), menustructuur en de daaraan gekoppelde autorisatie.

Ontwerp, bouw en test de gedeelten van de applicatie die niet onder rechtstreeks beheer van de applicatiegenerator vallen, zoals batch jobs en interfaces met andere systemen.

Ervaringen tot nu toe leren dat het realiseren van interfaces met andere systemen een lastig onderdeel blijft. Ofschoon veel afhangt van de technische infrastructuur en de technische mogelijkheden van de ontwikkelomgeving, verdient dit aspect ruim aandacht tijdens het ontwikkelproject. Het verdient aanbeveling hier direct na het bepalen van de systeemaafbakening aan te gaan werken, bijvoorbeeld door een speciaal daartoe vrijgemaakt projectteamlid.

Integreer alle systeemonderdelen van het increment en test het geheel (integratietest).

Eventueel kan het gerealiseerde, geïntegreerde systeem nog een keer worden gevalideerd met de gebruikersvertegenwoordigers en enkele eindgebruikers. Dit is aan te bevelen als de laatste gegenereerde systeemversie veel modificaties op de user interface heeft ondergaan. Vervolgens kan het systeem dan nog verder worden verfijnd totdat het de gebruikerswensen volledig afdekt. Echter, het aantal iteraties moet hier worden beperkt (bij voorkeur één, hooguit twee). Dit aantal kan op voorhand worden afgesproken met de stuurgroep.

9. Installeer het increment/systeem

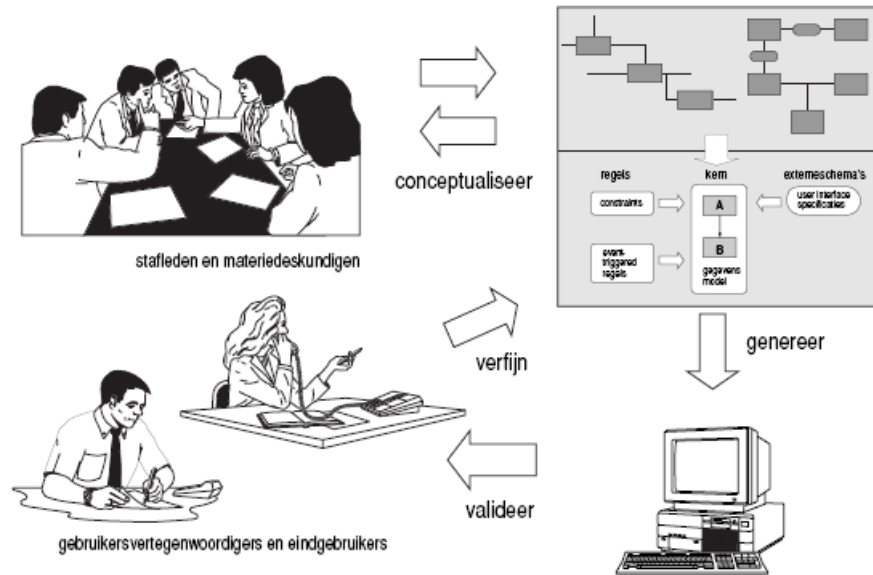
Integreer het increment in de gebruikers(test)omgeving. Voer de acceptatietest uit.

Gebruikersparticipatie

Figuur 31 illustreert de gebruikersparticipatie tijdens een MAD-project.

Stafleden en materiedeskundigen geven vorm aan het bedrijfsproces. Zij zetten de grote lijnen uit en bepalen de afbakening van de functionaliteit. De bedrijfsmodellen worden met hen afgestemd, bij voorkeur tijdens Joint Application Designsessies. Het gegenereerde prototype is hierbij een middel, zeker voor het afstemmen van het informatiemodel. Bij de definitie van de incrementen bepalen zij mede de prioriteitenvolgorde.

Gebruikersvertegenwoordigers bepalen – binnen de kaders van de bedrijfsmodellen – inhoudelijk de functionaliteit van het informatiesysteem, alsmede de user interface. Systemspecificaties worden met hen afgestemd, bij voorkeur in JADsessies aan de hand van gegenereerde systeemversies. Eventueel kan het definitieve informatiesysteem nog worden gevalideerd met de gebruikersvertegenwoordigers en enkele eindgebruikers.



Figuur 31 Gebruikersparticipatie

3.5 Onderhoud

We onderscheiden twee soorten onderhoud:

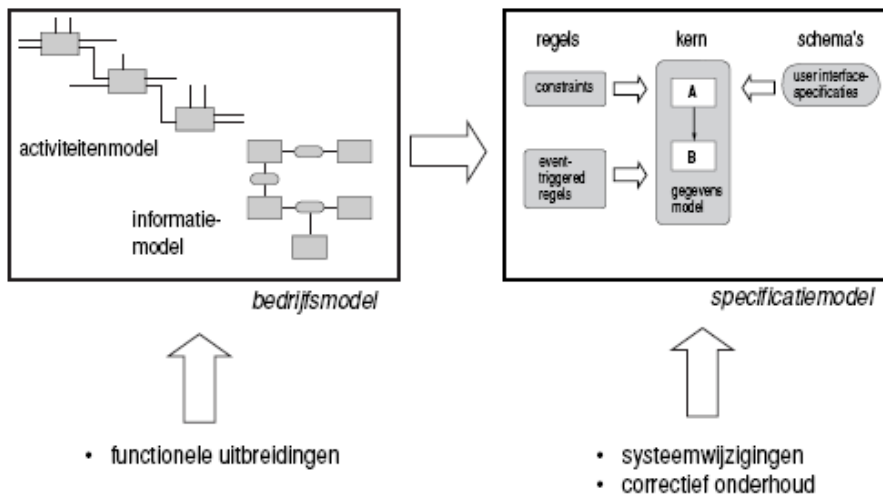
- functionele uitbreidingen;
- wijzigingen in het bestaande systeem.

Functionele uitbreidingen worden hetzelfde behandeld als een increment. De ontwikkeling hiervan is gelijk aan de incrementele ontwikkeling zoals hierboven beschreven. Uitbreiding van functionaliteit begint met het modelleren van bedrijfsactiviteiten (indien nodig) en het daaruit afleiden van het globale informatie(sub)model.

Systeemwijzigingen en correctief onderhoud worden uitgevoerd op het specificatieniveau en hebben betrekking op het gegevensmodel, de constraints, de eventtriggerde regels, rapportdefinities, batch jobs, user interface-specificaties en interfaces met andere systemen.

In relatie tot onderhoudbaarheid is het van belang het onderscheid tussen het bedrijfsmodel en het specificatiemodel te onderkennen. Het bedrijfsmodel is het resultaat van een bedrijfsanalyse en mag worden beschouwd als een architectuurmodel. Onderhoud van dit model is zinnig op het niveau van structuren. Dit geldt zeker voor het globale informatiemodel, waarin niet alle attribuuttypen en informatieregels in detail bekend hoeven te zijn. Het specificatiemodel daarentegen is het formele model van waaruit de applicatie wordt gegenereerd. Het specificatiemodel vormt voor de ontwikkelaar de 'broncode'. Onderhoud vindt op dat model plaats.

Figuur 32 illustreert het onderscheid tussen functionele wijzigingen enerzijds en systeemwijzigingen en correctief onderhoud anderzijds.



Figuur 32 Onderhoud

Het blijkt dat systemen die volgens MAD zijn ontwikkeld zeer efficiënt te onderhouden zijn. Dit komt omdat het systeemmodel gelijkvormig is met het bedrijfsmodel. Een verandering in de werkelijkheid wijst ondubbelzinnig naar de overeenkomstige plaats in het systeemmodel. Ten tweede is het ‘technisch ontwerp’ van het systeem impliciet. Iemand die de structuur van het gegevensmodel of het objectmodel kent, kent daarmee ook de structuur van de applicatie. Het systeem is immers te beschouwen als een executeerbaar model. Dit vergroot de toegankelijkheid voor onderhoudspersoneel aanzienlijk.

3.6 Samenvatting

In dit hoofdstuk zijn achtereenvolgens de planning, de uitvoering en de inrichting van een MAD-project behandeld.

In een contractuele relatie dient bij de planning van een MAD-project expliciet rekening te worden gehouden met het aspect van beheersing. Specificaties, opgetekend in een ‘allesomvattend’ mijlpaaldocument, ontbreken. Dit betekent dat een basis van een schatting, bijvoorbeeld uitgevoerd met functiepuntenanalyse, ontbreekt. Bovendien zal veelal nog niet precies duidelijk zijn wat de eisen en wensen ten aanzien van het te ontwikkelen systeem zijn.

Er wordt een iteratieve ontwikkelstrategie gevolgd, met systeemversies als prototypen, waarbij er geen andere verifieerbare tussenproducten worden gemaakt. Hierdoor kan de documentatie van de definitiefase aanzienlijk minder omvangrijk zijn dan bij een waterval-projectscenario, waarbij het gebruikelijk is dat ‘alle’ specificaties gedetailleerd op papier

worden gezet. Bovendien vraagt deze ontwikkelstrategie nieuwe instrumenten voor projectmanagement, zoals Time Box Management en het toekennen van prioriteiten aan functionaliteit.

De specificaties van het systeem worden opgesteld in intensieve groepsessies ('Joint Application Design') in plaats van in bilaterale interviews. Hierdoor worden tijdrovende heen-en-weer-afstemvergaderingen vermeden.

MAD-projecten worden uitgevoerd door kleine projectteams. Bij systeemontwikkeling volgens MAD komt de nadruk te liggen op twee soorten functies: de 'requirements-engineer' en de 'software-engineer'. De requirements-engineer is de architect, is communicatief, analytisch en conceptueel ingesteld, is in staat zich de door gebruikers aangedragen materiekennis snel eigen te maken en deze te verwoorden in modellen. De software-engineer is de ontwikkelaar, is vertrouwd met modelleren en vaardig met generatoren en de onderliggende techniek (Java, J2EE, Microsoft .NET, SQL, DBMS). Beiden werken dicht bij en nauw samen met de gebruikersorganisatie. Daarnaast staan specialisten op het gebied van database tuning, interface-protocol, GUI/ergonomie hun incidenteel ten dienste voor specifieke technische zaken.

Praktische ervaringen laten zien dat toepassing van MAD aanzienlijke verbeteringen in de productiviteit en de kwaliteit van systeemontwikkeling bewerkstelligen. Tevens heeft deze aanpak een stroomlijnend effect op de analyse en specificatie. De ervaringen leren ook dat onderhoud op deze systemen kan worden uitgevoerd op dezelfde productieve en kwalitatieve wijze.

De onderstaande tabel toont de gemeten productiviteit en snelheid van oplevering van een aantal MAD-projecten. Bij de berekening is uitgegaan van het volgende:

- De systeemomvang (in functiepunten) geldt inclusief interfaces met andere systemen.
- De project-productiviteitsratio geldt voor het gehele ontwikkeltraject (initiatie, definitie en realisatie), inclusief de tijd voor projectmanagement, projectteamvergaderingen, afstemsessies met gebruikers, training 'on-the-job', maar exclusief de uren van de gebruikers.
- Een mensmaand bevat 130 productieve uren.
- De snelheid van oplevering is berekend over de duur van het gehele project (van initiatie tot oplevering).

De projectomstandigheden zijn vergelijkbaar. In alle gevallen betrof het voor de projectleden een eerste kennismaking met MAD en waren zij 'onervaren' met de ontwikkelomgeving.

	Doorlooptijd in kalendermaanden	Omvang in functiepunten (FP)	Productiviteit Ontwerp en Realisatie in FP/dag	Project-inspanning in mensmaanden	Project-productiviteit in FP/mensmaand
Productiebesturings-systeem	12	1200	4	40	30
Financieel systeem	8	800	4	18	43
Management-informatiesysteem	5	600	8	11	54
Management-informatiesysteem*	5	1000	8	7	143

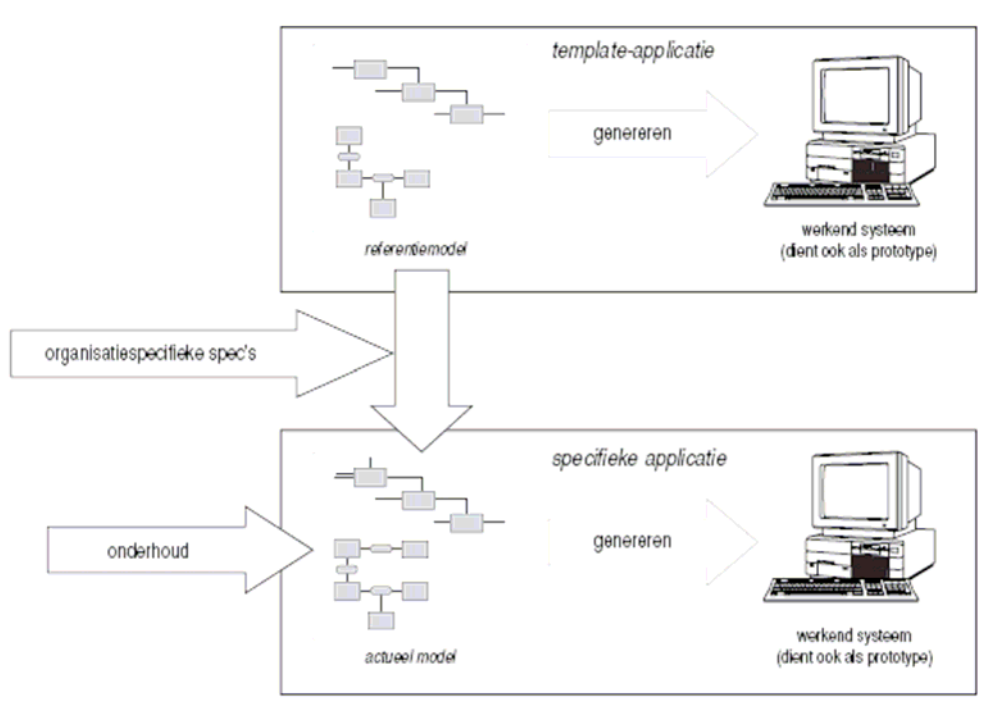
** Hierbij is gebruikgemaakt van de referentiemodellen van het hierboven vermelde management-informatiesysteem, verkregen van een andere, soortgelijke organisatie.*

De bovenstaande cijfers staan niet op zichzelf. Productiviteit en snelheid van opleveren zijn mede afhankelijk van de aard van de applicatie en de bekendheid van de organisatie – automatiseerders en gebruikers – met een projectaanpak volgens de MAD-werkwijze.

Opvallend is dat bij het hergebruik van specificaties c.q. referentiemodellen de productiviteit en snelheid van opleveren beduidend hoger ligt dan wanneer een project from scratch wordt uitgevoerd, startend met een bedrijfsanalyse. Op dit verschijnsel gaan we in het volgende hoofdstuk dieper in.

4 De template-benadering

In dit hoofdstuk schetsen we de template-benadering, een toepassing van MAD gebaseerd op het hergebruik van modellen of delen van modellen (zie figuur 33).



Figuur 33 De template-benadering

4.1 Definitie

De template-benadering wordt gekenmerkt door de aanwezigheid van:

- een applicatie-template voor een bepaald applicatiegebied;
- een werkende applicatie;
- een moderne ontwikkelomgeving om op eenvoudige wijze de applicatie aan te passen aan lokale, specifieke eisen en om te genereren naar een aantal verschillende serviceplatforms;
- een voorgeschreven methode voor aanpassing en implementatie van de applicatie.

We onderscheiden drie vormen:

- afdelingen binnen één organisatie gebruiken templates van elkaar;
- een leverancier hergebruikt een template bij steeds weer nieuwe klanten;
- de template is als commercieel product op de markt verkrijgbaar.

Andere namen voor applicatie-templates zijn ‘designware’, kernsystemen en rompsystemen.

Mogelijke toepassingen van de template-benadering zijn:

- voor gelijksoortige bedrijfsonderdelen binnen een bedrijf (bijvoorbeeld serviceorganisaties in diverse landen of fabrieken van een productdivisie)
- voor gelijksoortige problematiek voor meer bedrijven (bijvoorbeeld orderacceptatie, routingbepaling).

4.2 Hergebruik van specificaties

Een applicatie-template is een met specificaties gevulde repository van waaruit een werkende applicatie wordt gegenereerd. Een referentiemodel kan aan de template ten grondslag liggen. Dit model is algemeen van karakter (bijvoorbeeld financieel, logistiek) of bedrijfstakspecifiek (bijvoorbeeld verzekeringen). Het referentiemodel is een weergave van het bedrijfsproces binnen het applicatiegebied.

Wanneer de applicatie-template en het hieraan ten grondslag liggende referentiemodel beschikbaar worden gesteld, spitst het hergebruik zich op twee niveaus toe:

- het bedrijfsmodel;
- het specificatiemodel.

Het bedrijfsmodel bevat een beschrijving van de bedrijfsomgeving en omvat het bedrijfsactiviteitenmodel en het informatiemodel. Het specificatiemodel bevat de systeemspecificaties gezien vanuit een systeemtechnische invalshoek en bestaat uit de volgende componenten:

- het relationele gegevensmodel (in het geval van een relationele database);
- constraints en event-getriggerde regels;
- programmodulen en schermdefinities;
- een menustructuurdefinitie.

4.3 Moderne ontwikkelomgeving

Aanpassing van de template vindt plaats in een moderne ontwikkelomgeving. CASE-tools ondersteunen een standaardwijze van werken en bewerkstelligen een hoge kwaliteit en een hoge productiviteit. De applicatie kan worden gegenereerd voor verschillende hardware- en softwareplatforms. We onderscheiden twee soorten tools:

- CASE-tools die bedrijfsmodellering ondersteunen;
- applicatiegeneratoren.

De CASE-tools die bedrijfsmodellering ondersteunen, worden vaak aangeduid als ‘upper-CASE’, terwijl de generatoren worden aangeduid als ‘lower-CASE’.

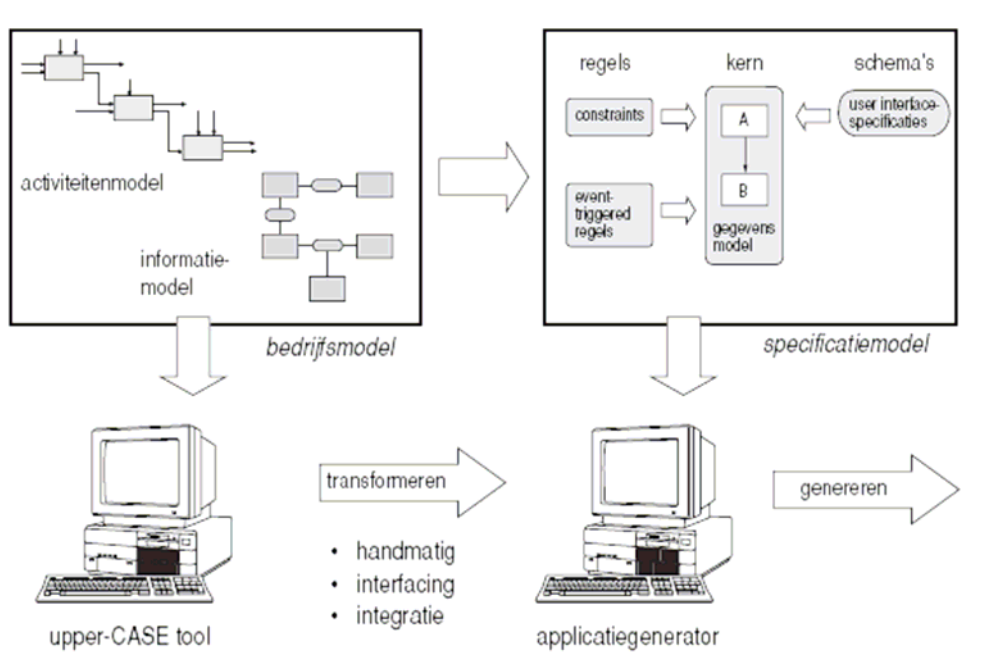
Het bedrijfsmodel zal doorgaans gedocumenteerd zijn met behulp van een upper-CASE tool. De specificatiemodellen van waaruit gegenereerd wordt, zullen opgeslagen zijn in de repository van de applicatiegenerator. Er zijn drie manieren om deze te vullen:

- handmatig;
- via een brug, een interface-programma van de ‘upper-CASE’ tool naar de applicatiegenerator;
- geautomatiseerd, wanneer er gebruik wordt gemaakt van een geïntegreerde ontwikkelomgeving.

Figuur 34 toont de relatie tussen enerzijds het bedrijfsmodel en de opslag hiervan in een upper-CASE tool en anderzijds het specificatiemodel en de opslag hiervan in de repository van de applicatiegenerator.

Integratie van de hulpmiddelen wordt bereikt via koppelprogramma’s (‘interfacing’) of door gebruik van een gemeenschappelijke repository (‘integration’). Beide vormen worden aangeduid met de term I-CASE.

Overigens is het de vraag of de referentiemodellen wel in de vorm van een gevulde repository van een CASE-tool beschikbaar moeten worden gesteld. Als er bij de ontvangende organisatie al een ontwikkelomgeving (lees: een echte applicatiegenerator) beschikbaar is, is het verkrijgen van de modellen in een CASE-formaat triviaal. Wat dan nog moet worden gedaan is het overzetten van het referentiemodel op specificatieniveau naar de repository van de eigen applicatiegenerator. Ervaringen hebben geleerd dat dit weinig inspanning vergt en dat het soms zelfs wenselijk is om dit ‘handmatig’ te doen, omdat de ontwikkelaar dan pas een goed begrip krijgt van het specificatiemodel.



Figuur 34 CASE-tools

4.4 Acquisitie van de template

Er is een aantal mogelijkheden om een applicatie-template te verkrijgen, waarvan we de volgende noemen:

- van aanbieders op de markt. De eerste commerciële aanbieders van applicatietemplates dienen zich aan. Hierbij kunnen de volgende groepen worden onderscheiden:
 - CASE-tool-leveranciers. Sommige aanbieders van CASE-tools opereren als makelaar op de templates-markt. Andere bieden hun modellen aan met de daaraan gerelateerde consultancy-expertise als extra verkoopargument voor hun producten;
 - leveranciers van softwarepakketten;
 - management-consultants.
- uit eerder uitgevoerde projecten:
 - door ‘klassieke’ specificaties (logische en technische ontwerpen) om te zetten naar ‘template’-specificaties: het gegevensmodel en de hierop afgebeelde constraints en event-getriggerde regels;
 - bij de migratie/re-development van een bestaand systeem door de databasestructuur weer ‘af te beelden’ naar een logisch gegevensmodel en dit via een aantal iteraties verder uit te bouwen;
 - door initieel een applicatie op te zetten als een template, wat in feite neerkomt op het analyseren en specificeren volgens de in paragraaf 3.4 geschetste werkwijze, waarbij de modellen volgens de in hoofdstuk 2 beschreven modellencyclus worden opgezet.

door uitwisseling tussen organisaties.

4.5 Aanpassen en invoeren van de template

Afhankelijk van de aard van de applicatie en de gewenste functionaliteit van de organisatie zal een belangrijk deel van de functionaliteit worden afgedekt door de applicatie-template. Het te veranderen of ontbrekende deel wordt gewijzigd of toegevoegd op specificatieniveau. Voor het aanpassen van de template is een standaardwerkwijze beschikbaar. In het algemeen bestaat deze werkwijze uit de volgende stappen:

- een verschillenanalyse;
- aanpassing van het specificatiemodel;
- constructie van het specifieke systeem;
- implementatie van het specifieke systeem.

Als eerste stap wordt onderzocht in hoeverre de aan de template ten grondslag liggende referentie- of bedrijfsmodellen de specifieke werkwijze van de organisatie afdekken. Hiertoe kan men een bedrijfsanalyse uitvoeren met als doel de gewenste werkwijze van de organisatie in kaart te brengen. De resultaten hiervan worden gedocumenteerd in een specifiek bedrijfsmodel. Vervolgens wordt nagegaan in hoeverre het bedrijfsspecifieke model afwijkt van het referentiemodel van de applicatie-template.

Een andere weg die men kan bewandelen is het bespreken van de in de template opgenomen modellen met de gebruikersvertegenwoordigers. Hierbij wordt een vergelijking gedaan van model en gewenste werkwijze. Een gegenereerd prototype kan hierbij van dienst zijn. Mogelijke nadelen van de laatste optie zijn dat de creativiteit wordt onderdrukt en dat een organisatie de modellen moeilijk accepteert vanwege het ‘not invented here’-syndroom.

In het gebruik van een applicatie-template voor het genereren van een eerste systeemversie schuilt een extra risico. Het eenvoudigweg ‘overnemen’ van een specificatiemodel zonder dat de architecten van dat model worden ‘meegeleverd’ is geen goede werkwijze. Modellen zijn per definitie abstract, en ontwerpkeuzen die in het model zijn gemaakt zijn moeilijk documenteerbaar. Hierdoor krijgen ‘onbekenden’ niet snel inzicht in de modellen. Het verdient daarom aanbeveling om bij de overname van een applicatie-template ook consultancy van de opstellers ervan te betrekken, zodat het nieuwe team in korte tijd kan worden ingewerkt in de template.

Het resultaat is in beide gevallen een verschillenlijst.

Vervolgens wordt het specificatiemodel op basis van de geconstateerde verschillen aangepast en uitgebreid. Joint Application Design en prototyping worden toegepast om samen met de gebruikers de aanvullende eisen en wensen boven water te krijgen. De constructie van het specifieke systeem bestaat uit het genereren van de applicatie op basis van het specifieke specificatiemodel. De systeemconstructie behelst tevens het eventueel aanpassen van de user interface conform lokale cosmetische eisen, het toevoegen van interfaces naar andere systemen en het toevoegen van specifieke rapporten. De implementatie van het specifieke systeem omvat de conversie naar het nieuwe systeem en de training van de gebruikers.

Onderhoud

Functionele veranderingen, resulterend in een uitbreiding van entiteitstypen of attriboottypen, worden eerst aangebracht in het bedrijfsmodel en vervolgens doorgevoerd naar het specificatiemodel, van waaruit vervolgens de applicatiecode wordt gegenereerd. Met het oog op onderhoudbaarheid dienen veranderingen direct in de gegenereerde code te worden vermeden. Systeemveranderingen en correctief onderhoud vinden direct plaats op het niveau van het specificatiemodel, resulterend in aanpassingen op het gegevensmodel, regels, rapporten, batch jobs, user interface-modificaties en interfaces naar andere systemen.

Het onderhoud kan centraal of decentraal worden georganiseerd. Bij centraal onderhoud zal men de template zo compleet mogelijk maken. Diversificatie wordt bewerkstelligd door de bedrijfsregels van de verschillende bedrijfsonderdelen in de repository ‘aan- en uitzetbaar’ (parametriseerbaar) te maken. Bij het genereren van een versie voor een bepaald bedrijfsonderdeel worden alleen de voor die organisatie van toepassing zijnde bedrijfsregels meegenomen. Aanvullende wensen van verschillende bedrijfsonderdelen worden centraal gerealiseerd.

Het centraal onderhoud is in het bijzonder geschikt wanneer men vanuit ‘de centrale’ een uniforme werkwijze bij alle bedrijfsonderdelen min of meer wil ‘afdwingen’. Bij decentraal onderhoud draagt het ontvangende bedrijfs onderdeel zelf zorg voor het onderhoud en het realiseren van additionele wensen. Bij deze vorm van gedistribueerd onderhoud heeft elk bedrijfs onderdeel de benodigde expertise hiervoor nodig.

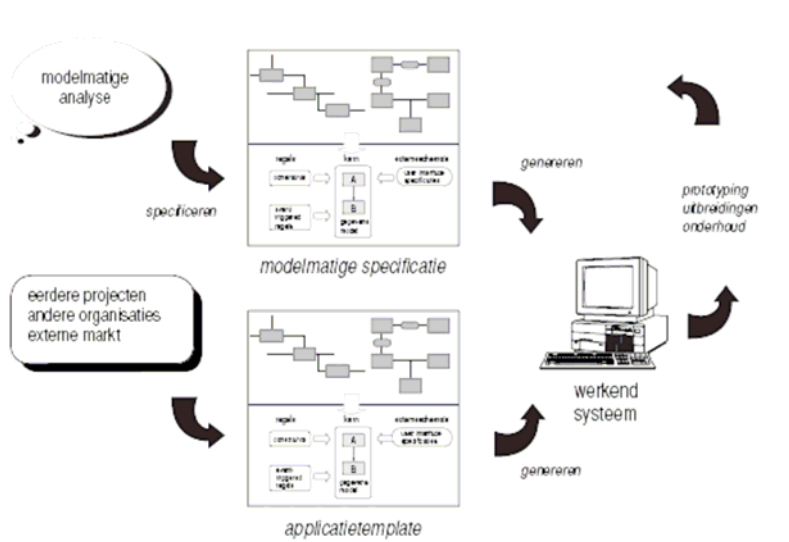
4.6 Samenvatting

In dit hoofdstuk is de template-benadering als toepassing van MAD gebaseerd op het hergebruik van modellen toegelicht. Het hergebruik van specificaties is geënt op referentiemodellen voor een specifiek applicatiegebied. Aanpassing vindt plaats in een moderne ontwikkelomgeving. CASE-tools ondersteunen een standaardwijze van werken en bewerkstelligen een hoge kwaliteit en een hoge productiviteit. De applicatie kan worden gegenereerd voor verschillende hardware- en softwareplatforms.

Applicatie-templates, met name de referentiemodellen, geven een organisatie een vliegende start voor de bedrijfsanalyse en specificatie van het informatiesysteem.

De gevolgde werkwijze bij het aanpassen en invoeren van de template verschilt nauwelijks van de in paragraaf 3.4 geschetste werkwijze. Het enige verschil is dat op voorhand reeds een bedrijfs- of specificatiemodel beschikbaar is. Figuur 35 illustreert de integratie van beide werkwijzen:

- zelfbouw, startend met een bedrijfsanalyse, eventueel aansluitend op een informatie-plannings- of BPR-traject, waarbij een blauwdruk van het bedrijfsof procesmodel voorhanden is;
- zelfbouw, waarbij gebruik wordt gemaakt van het reeds beschikbare referentiemodel of van een in een repository opgeslagen applicatie-template. Deze kunnen uit verschillende bronnen worden verkregen: van een ander soortgelijk project, van een andere organisatie of van een leverancier.



Figuur 35 Zelfbouw gecombineerd met referentiemodellen

5 De Actimod-methode

De Actimod-methode is afgeleid van SADT – Structured Analysis and Design Technique [Marca] – en is bij Philips ontwikkeld als methode voor het modelleren van bedrijfsactiviteiten. Momenteel is deze methode ook bekend als IDEF0.

5.1 Algemene concepten

Een activiteitenmodel volgens Actimod beschrijft de bedrijfsactiviteiten met hun informatie- en materiestromen met behulp van samenhangende activiteitendiagrammen met bijbehorende beschrijvingen. De activiteitendiagrammen vormen samen als het ware een piramide waarvan elke laag de activiteiten op een bepaald niveau van detail beschrijft. Hoe lager in de piramide, hoe groter de mate van detail.

Met Actimod worden de bedrijfsactiviteiten los van hun huidige of wellicht toekomstige organisatorische inpassing gemodelleerd, dit om van elke activiteit slechts de werkelijk relevante informatiebehoefte en informatieproductie vast te kunnen stellen. Het modelleren van organisatorische eenheden zou dit beeld mogelijkwijs vertroebelen. Zo zal ‘orders verwerken’ wel een juist gemodelleerde activiteit zijn en ‘orderafdeling’ niet. Het zorgvuldig naleven van de naamgevingconventies biedt hierbij overigens ondersteuning. Het daarnaast gebruiken van bijvoorbeeld organisatieschema’s is natuurlijk niet verboden. Alles wat het inzicht in de bedrijfsactiviteiten verhoogt, kan nuttig zijn.

Actimod kent de volgende begrippen:

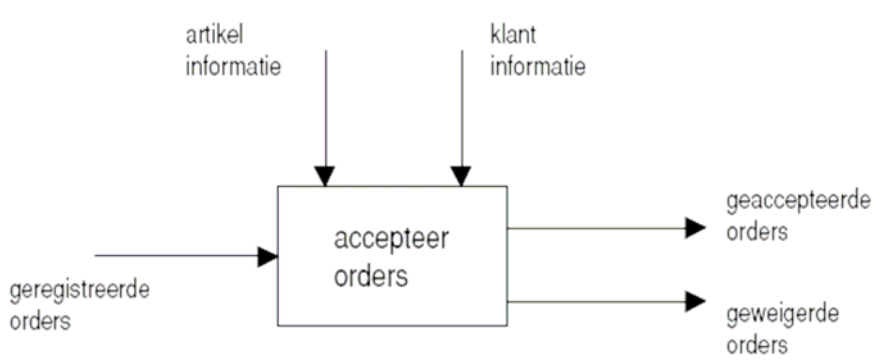
- activiteit: een bedrijfsactiviteit uitgevoerd door mens of machine;
- informatiestroom: uitwisseling van informatie tussen activiteiten;
- materiestroom: uitwisseling van materie tussen activiteiten.

De grafische syntaxis bestaat uit een rechthoek voor activiteiten en pijlen voor materie- en informatiestromen. Voor de namen van activiteiten worden werkwoorden gebruikt. Materie- en informatiestromen worden aangeduid met zelfstandige naamwoorden. Actimod kent een onderscheid tussen invoer-, uitvoer- en controlestromen (‘input’, ‘output’ en ‘control’):

- Een invoerstroom ondergaat een transformatie.
- Een uitvoerstroom is het resultaat van die transformatie.
- Een controlestroom wordt niet getransformeerd, maar wordt gebruikt of geraadpleegd om de uitvoerstroom te kunnen produceren.

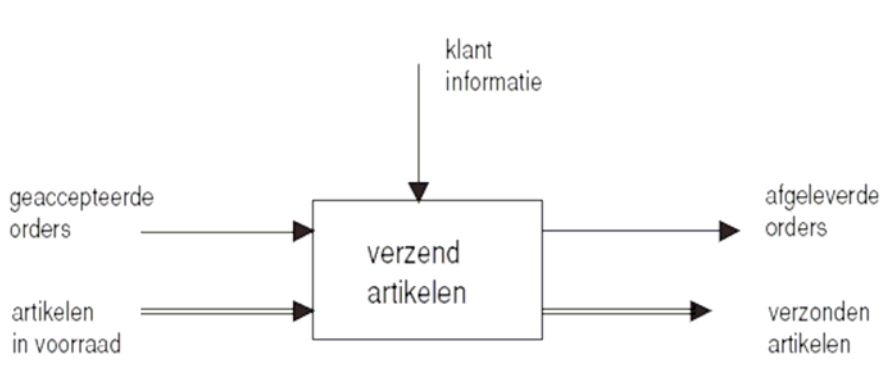
In de grafische syntaxis komen invoerstromen aan de linkerkant van de activiteit binnen, controlestromen komen aan de bovenkant binnen en uitvoerstromen verlaten de activiteit aan de rechterkant.

Figuur 36 toont een bedrijfsactiviteit met één invoerstroom, twee controlestromen en twee uitvoerstromen. De controlestromen Artikelinformatie en Klantinformatie ondergaan geen verandering. De informatie uit deze stromen wordt alleen geraadpleegd om te kunnen bepalen of de order kan worden geaccepteerd. De invoerstroom Geregistreerde order ondergaat wel een verandering. De orderstatus wordt gewijzigd van ‘geregistreerd’ in ‘geaccepteerd’ of ‘geweigerd’.



Figuur 36 Invoer-, controle- en uitvoerstromen

De grafische syntaxis van Actimod kent tevens een onderscheid tussen materie- en informatiestromen (figuur 37). Informatiestromen worden getekend als een enkele pijl. Materiestromen worden getekend als een dubbele pijl. Het opnemen van materiestromen kan de herkenbaarheid van het model, vooral voor de gebruikers, ten goede komen.



Figuur 37 Materie- en informatiestromen

Naamgeving

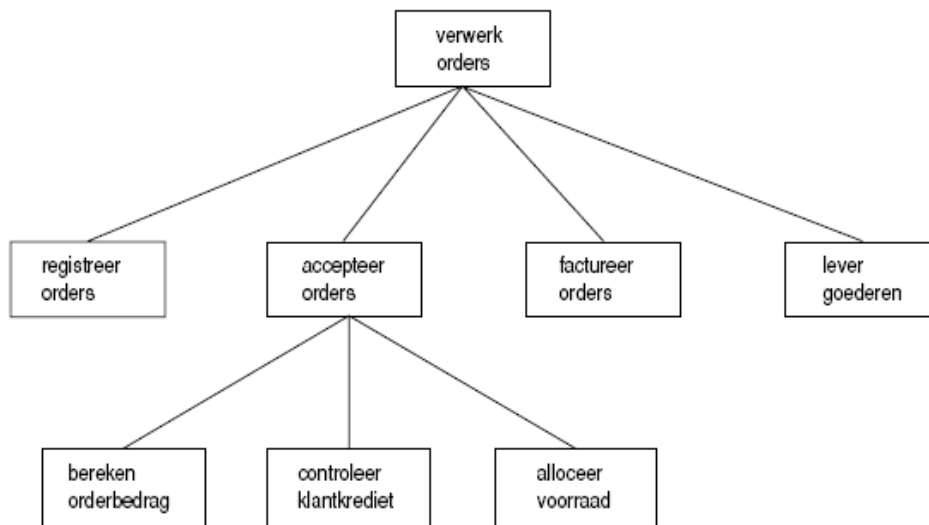
Zoals al eerder vermeld worden bedrijfsactiviteiten aangeduid met werkwoorden (eventueel aangevuld met een zelfstandig naamwoord) en materie- en informatiestromen met zelfstandige naamwoorden. Voorbeelden van goede namen voor activiteiten zijn: Verzenden goederen, Controleer voorraad en Accepteer order. Het is beter om de namen voor activiteiten ‘uitvoer-gericht’ te kiezen dan ‘invoergericht’. Zo is het beter om te spreken van ‘Accepteer order’ in plaats van ‘Ontvang order’.

5.2 Decompositie

Actimod maakt – zoals vele procesgeoriënteerde methoden – gebruik van decompositie. Het decompositieproces is recursief. Elke op een lager niveau voorkomende bedrijfsactiviteit kan weer worden ontleed in een aantal kleinere activiteiten. Dit proces wordt voortgezet totdat het gewenste niveau van detaillering is bereikt.

Decompositie van activiteiten

In het voorbeeld uit figuur 38 wordt gestart met één activiteit. Deze activiteit representeert het applicatiegebied dat nader wordt geanalyseerd. Vervolgens wordt deze activiteit ontleed in een aantal subactiviteiten. De subactiviteiten dekken samen de activiteit van het hogere niveau af. In principe kan elke subactiviteit nu verder worden ontleed om meer inzicht te verkrijgen in de werking en informatie-uitwisseling van die activiteit. In het voorbeeld van figuur 38 is dit beperkt tot één subactiviteit.



Figuur 38 Decompositie

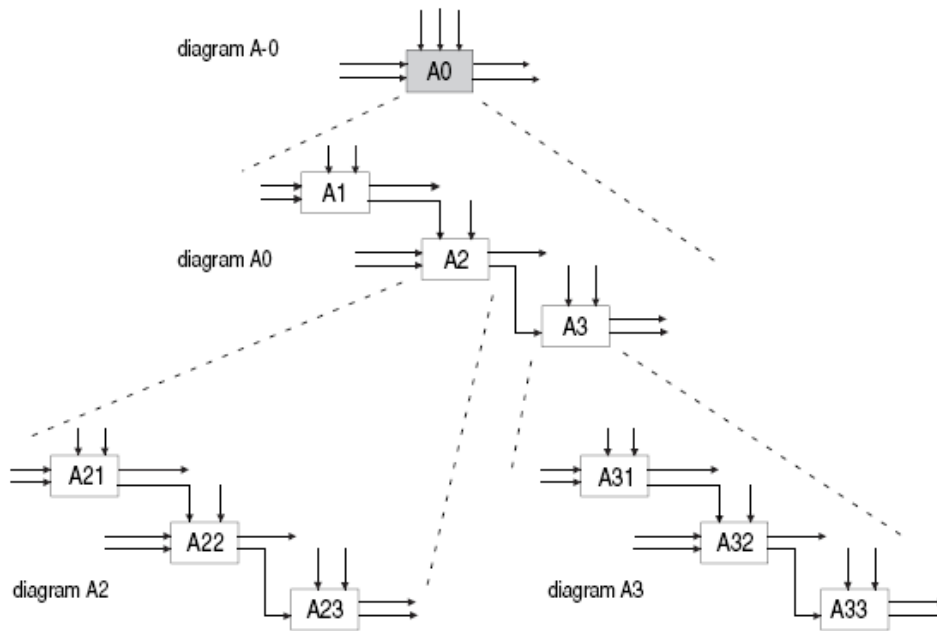
Bij het ontleden van activiteiten moet men streven naar een gelijkmatige verdeling ('horizontal balancing'). De activiteiten die op het nieuwe diagram ontstaan moeten met elkaar in evenwicht zijn.

Decompositie van activiteiten stopt op het niveau waarop kan worden bepaald of een activiteit volledig handmatig wordt uitgevoerd of volledig formaliseerbaar is. Wanneer een activiteit beide componenten heeft, dus zowel handmatige als formaliseerbare aspecten, wordt deze activiteit een niveau dieper ontleed.

Identificatie van diagrammen en activiteiten

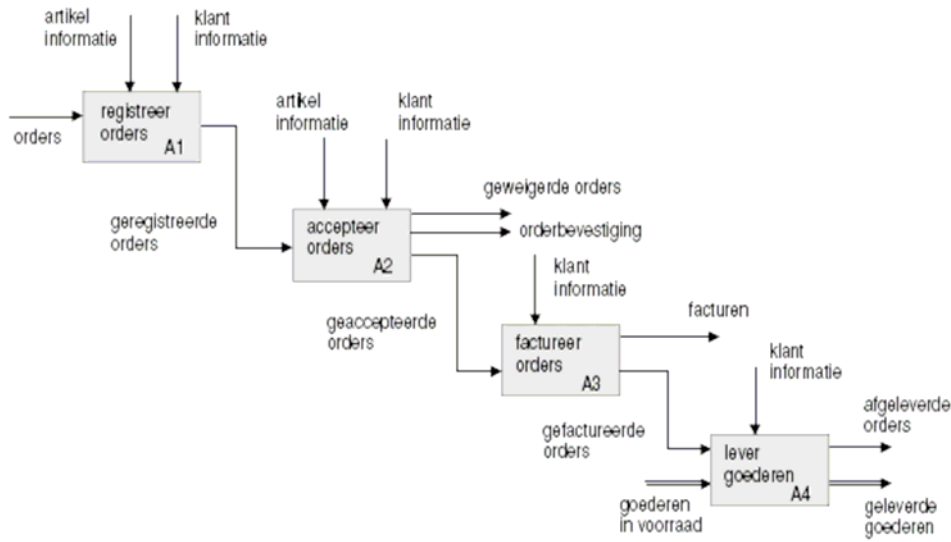
Figuur 39 toont de identificatie voor diagrammen en activiteiten. Het diagram A0 (lees: A min nul) toont één activiteit, die wordt geïdentificeerd als A0. Deze activiteit wordt verder

ontleed in diagram A0, dat een aantal activiteiten bevat. Deze activiteiten worden genummerd als A1, A2, A3, enzovoort. In het voorbeeld van figuur 39 worden de activiteiten A2 en A3 verder gedetailleerd in respectievelijk diagram A2 en diagram A3. Diagram A2 toont de activiteiten A21, A22 en A23, diagram A3 toont de activiteiten A31, A32 en A33.

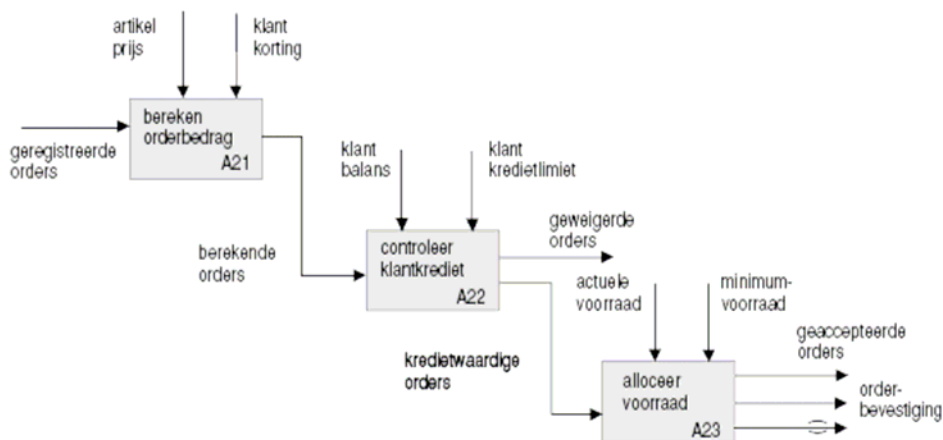


Figuur 39 Naamgevingconventies bij decompositie

Figuur 40 toont het diagram A0 van een orderverwerkingsproces. Dit diagram toont vier hoofdactiviteiten, waarvan activiteit A2 verder is gedetailleerd in diagram A2 (zie figuur 41).



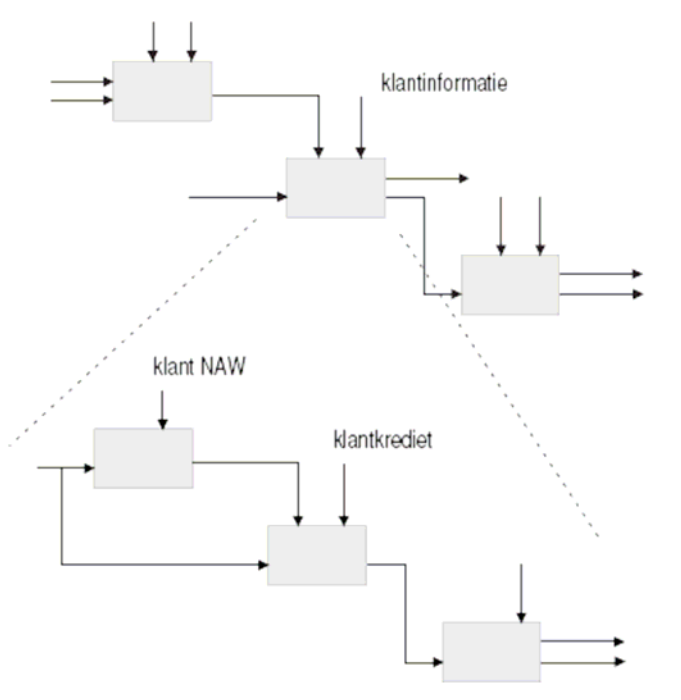
Figuur 40 Diagram A0: Verwerk orders



Figuur 41 Diagram A2: Accepteer orders

Decompositie van materie- en informatiestromen

Naast het ontleden van activiteiten heeft men ook de mogelijkheid om materie- en informatiestromen te ontleden. Figuur 42 geeft hiervan een voorbeeld. In dit voorbeeld wordt de informatiestroom Klant informatie op een lager niveau ontleden in twee deelstromen die elk op dat lagere niveau een verschillende activiteit bedienen. Hierdoor heeft de modelleur de mogelijkheid om een activiteit enkel en alleen die informatie te geven die noodzakelijk is voor het vervullen van zijn taak.

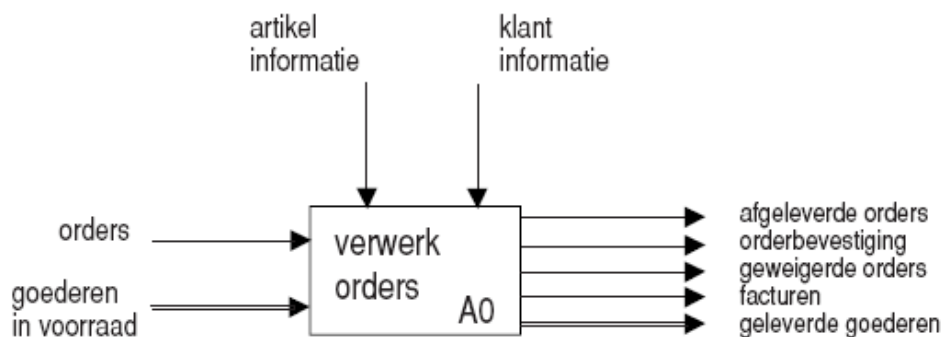


Figuur 42 Decompositie van een informatiestroom

5.3 Conventies

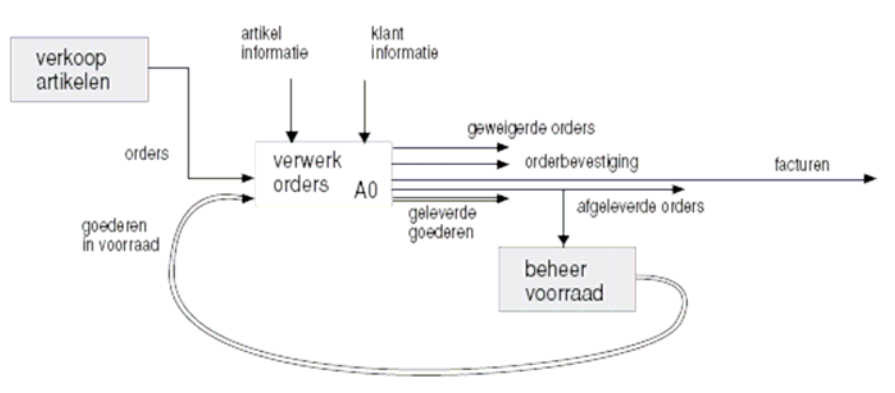
Het top- en het contextdiagram

De afbakening van het applicatiegebied wordt aangegeven op het topdiagram (genaamd A-0). Figuur 43 toont hiervan een voorbeeld. Het topdiagram bevat één activiteit, die vervolgens verder wordt gedetailleerd in een aantal subactiviteiten.



Figuur 43 Het topdiagram voor Verwerk orders

Wanneer men ook de samenhang van het applicatiegebied met zijn directe omgeving wil aangeven, kan men daartoe een contextdiagram tekenen. Het wordt genoemd A-1 diagram (lees: A min één diagram). Dit diagram toont ook activiteiten in de naaste omgeving waarmee de activiteit van het te analyseren applicatiegebied materie en informatie uitwisselt. Figuur 44 toont een voorbeeld van een contextdiagram.



Figuur 44 Het contextdiagram voor Verwerk orders

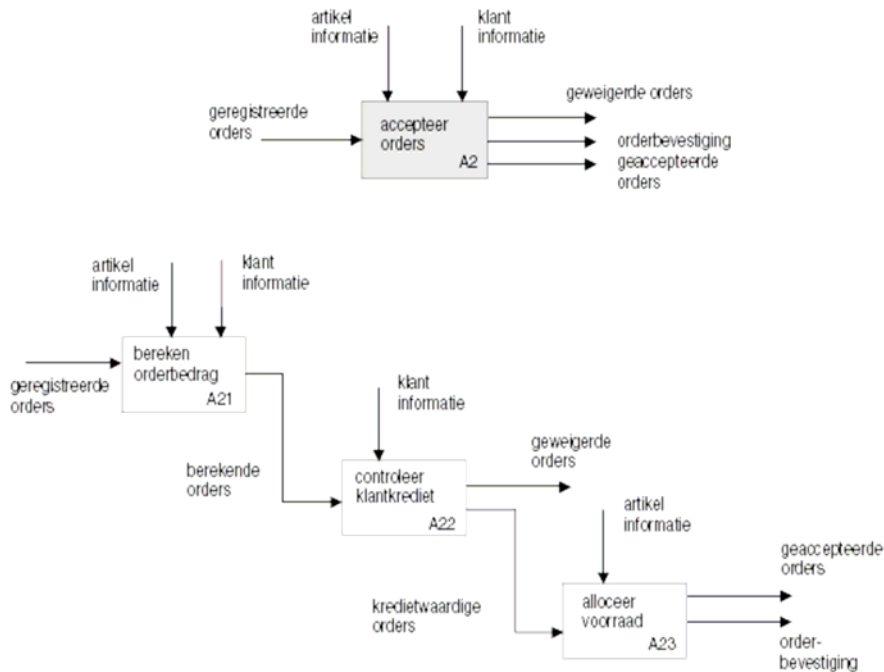
Consistentie tussen diagrammen

Wanneer een activiteit wordt ontleed, moet de som van alle ingaande en uitgaande stromen van de decompositie gelijk zijn aan de ingaande en uitgaande stromen van die activiteit en vice versa ('level balancing'). De informatiestromen tussen de activiteiten op de decompositie blijven buiten beschouwing.

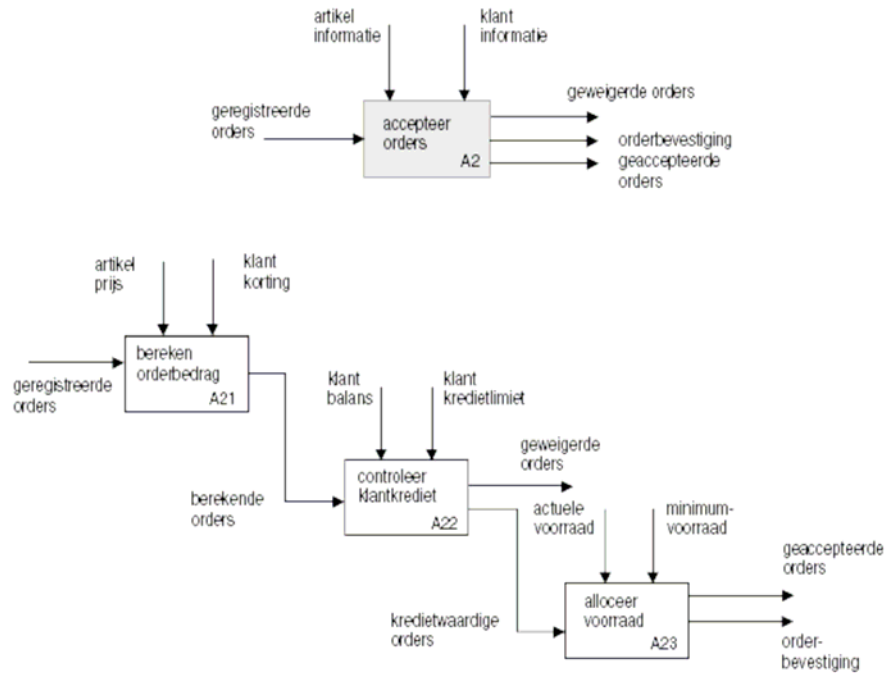
Figuur 45 illustreert dit. In dit voorbeeld is de decompositie in balans met activiteit A2: alle 'losse' ingaande en uitgaande informatiestromen op het diagram komen ook voor als ingaande en uitgaande informatiestromen van activiteit A2 en vice versa.

Als ook stromen op een lager niveau worden ontleed, moeten de diagrammen nog steeds onderling consistent blijven. In dit geval is het raadzaam om de decompositie van stromen te vermelden. Dit kan op het diagram of op een apart tekstblad.

Figuur 46 geeft hiervan een voorbeeld. In dit voorbeeld is de informatiestroom Artikelinformatie op het lagere niveau ontleed in de informatiestromen Artikelprijs, Actuele voorraad en Minimumvoorraad. De informatiestroom Klant informatie is ontleed in de informatiestromen Klantkorting, Klantbalans en Klantkredietlimiet.



Figuur 45 Consistentie tussen diagrammen

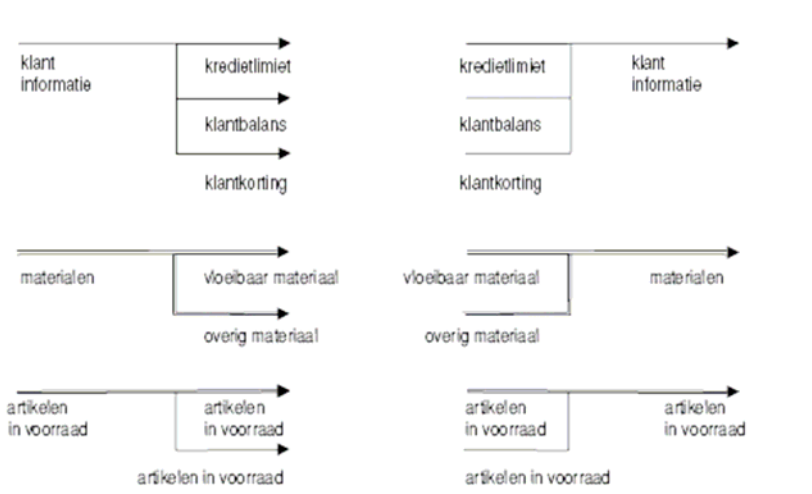


Figuur 46 Consistentie tussen diagrammen bij het ontleden van stromen

Klantkorting, Klantbalans en Klantkredietlimiet zijn deelstromen van Klant informatie. Artikel prijs, Actuele voorraad en Minimumvoorraad zijn deelstromen van Artikel informatie.

Splitsen en samenvoegen

Stromen kunnen worden gesplitst in deelstromen ('branching') en deelstromen kunnen worden samengevoegd tot één stroom ('joining'). Figuur 47 illustreert dit voor materiële stromen, informatiestromen en een combinatie hiervan.



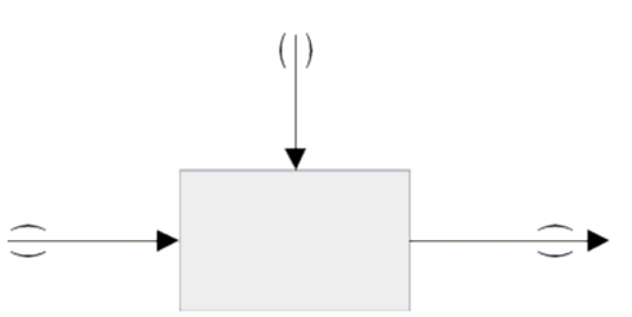
Figuur 47 Splitsen en samenvoegen van materie- en informatiestromen

Introduceren van nieuwe stromen

Meestal is een informatiestroom op een lager niveau één van de volgende stromen:

- dezelfde stroom als de stroom op het hogere niveau;
- een decompositie van een stroom op het hogere niveau;
- een stroom tussen activiteiten binnen het diagram (op het lagere niveau).

Toch is het mogelijk om een totaal nieuwe stroom op een lager niveau te introduceren ('tunnelling'). Deze stromen zijn niet relevant voor de hoger gelegen niveaus. Op het diagram waar deze stromen worden geïntroduceerd, worden ze aangegeven met haakjes. Figuur 48 illustreert dit. De introductie van nieuwe stromen geldt voor invoer-, controle- en uitvoerstromen.



Figuur 48 Introductie van nieuwe stromen

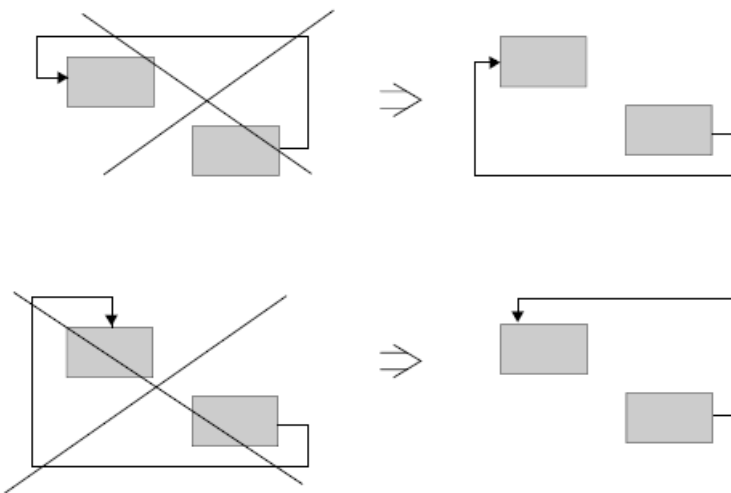
De stromen die op deze wijze nieuw worden geïntroduceerd worden niet gecontroleerd op consistentie met het diagram van een hoger niveau. Het gebruik van het introduceren van nieuwe stromen dient daarom beperkt te blijven.

Terugkoppeling

Binnen een diagram zijn er twee mogelijkheden om informatiestromen terug te koppelen naar een andere activiteit:

- van uitvoer naar invoer;
- van uitvoer naar controle.

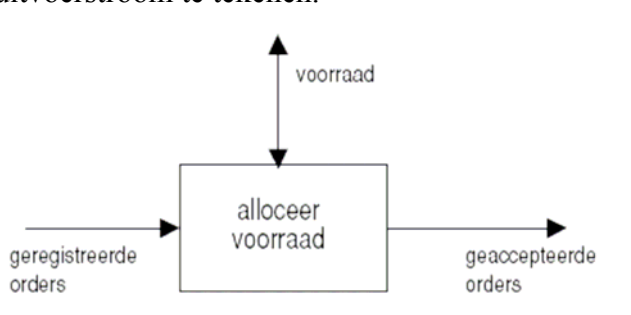
Stromen van uitvoer naar invoer worden onderlangs getekend. Stromen van uitvoer naar controle worden bovenlangs getekend. Figuur 49 illustreert dit.



Figuur 49 Terugkoppeling

Combinatie van controle en uitvoer

Een controlestroom kan direct geactualiseerd worden. Figuur 50 laat de tekenwijze hiervoor zien. In dit geval is het niet nodig om de controlestroom Voorraad ook nog als uitvoerstroom te tekenen.



Figuur 50 Controle-/uitvoerstroom

6 De Entity-Relationship-methode

Helaas is er niet zo iets als de Entity-Relationship-methode (ER-methode). Er zijn diverse varianten, zowel wat tekentechnieken als wat concepten betreft. Wij hebben in dit boek gekozen voor de ISO-variant [ISO]. De ER-methode wordt vaak geassocieerd met ‘klassieke’ gegevensmodellering. Toch komen enkele concepten en diagramtechnieken hieruit ook terug in objectgeoriënteerde methoden. Dat is niet zo verwonderlijk wanneer men bedenkt dat in het modelleren van een bedrijfswerkelijkheid veel overeenkomsten bestaan tussen de ‘klassieke’ informatiemodellering en objectoriëntatie. Unified Modelling Language (UML) is voor een deel gebaseerd op de ER-methode.

In paragraaf 6.1 beschrijven we eerst de algemene concepten van de ER-methode. Vervolgens wordt in paragraaf 6.2 de ISO-variant beschreven. Tot slot geven we in paragraaf 6.3 enkele aanvullingen.

6.1 Algemene concepten

In 1976 publiceerde Peter Pin-shan Chen van het Massachusetts Institute of Technology zijn inmiddels beroemde artikel ‘The Entity-Relationship Model – Toward a Unified View of Data’. Dit artikel wordt gezien als de eerste officiële publikatie over de ER-methode.

Sinds het artikel van Chen zijn er vele varianten van ER ontstaan. Ondanks deze varianten kan men zeggen dat de ER-methode de volgende algemene concepten kent:

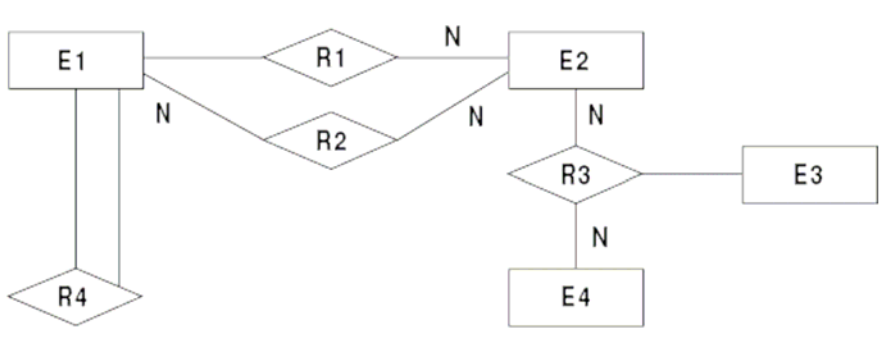
- Entiteit: een object uit het universum van discussie.
Attribuut: een eigenschap van een entiteit of een associatie tussen entiteiten.
Relatie: een associatie tussen entiteiten.
- Typen worden gebruikt als classificatie van deze concepten.
- De grafische notatie bestaat uit een rechthoek voor een entiteitstype, een ruit en verbindende lijnen voor een relatietype. In de symbolen worden namen geschreven.
- N geeft de aard van het relatietype aan.

Figuur 51 illustreert de grafische notatie. Rechthoeken representeren entiteitstypen. E1, E2, E3 en E4 zijn de namen van de entiteitstypen. Voor de namen van de entiteitstypen worden zelfstandige naamwoorden gebruikt.

Ruiten met verbindende lijnstukken representeren relatiestypen. R1, R2, R3 en R4 zijn de namen van de relatiestypen. Voor de namen van de relatiestypen worden werkwoorden gebruikt. R1 en R2 zijn binaire relatiestypen. R3 is een ternair relatietype. R4 is een recursief (‘involved’) relatietype. N geeft de aard van het relatietype aan. Wanneer er geen ‘N’ is aangegeven, wordt aangenomen dat deze ‘1’ is.

Een entiteit-relationship-diagram, zoals figuur 51, toont de entiteitstypen en de relatiestypen. De attribuuttypen van entiteitstypen worden toegevoegd in een tekstuele beschrijving. De

identificerende attribuuftypen maken deel uit van deze beschrijving. Een voorbeeld is hieronder gegeven. Hierin is E1 de naam van het entiteittype, I1 het identificerende attribuuftype ervan en A1 en A2 de overige attribuuftypen ervan.



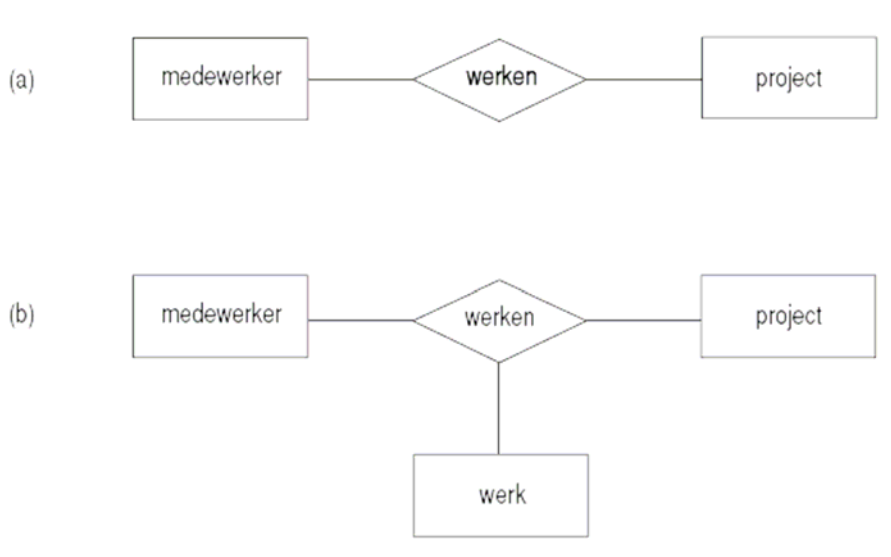
Figuur 51 Algemeen concept van ER

Entiteitstypen en relatie-typen

Binnen de concepten van Chen's oorspronkelijke model is er geen fundamenteel onderscheid tussen entiteitstypen en relatie-typen. Een relatie-typen kan vaak ook worden beschouwd als entiteitstypen. Bij het model van Chen kunnen zowel entiteitstypen als relatie-typen attribuuftypen bezitten. Of een bepaald feit wordt gemodelleerd als entiteitstypen of als relatie-typen, hangt af van de voorkeur van de modelleur.

Een voorbeeld hiervan is gegeven in figuur 52. In figuur 52(a) wordt het feit dat een medewerker aan een project werkt door de modelleur gezien als een relatie-typen. Dit relatie-typen heeft als attribuuftypen 'begindatum' en 'duur'. In figuur 52(b) ziet de modelleur hetzelfde feit als een entiteitstypen. Hier heeft het entiteitstypen 'Werk' de attribuuftypen 'begindatum' en 'duur'. In dit laatste geval heeft het relatie-typen geen attribuuftypen.

Figuur 52(a) toont een binair relatie-typen. Dit is een relatie-typen waarbij twee entiteitstypen betrokken zijn. Figuur 52(b) laat een ternair relatie-typen zien. Hierbij zijn drie entiteitstypen betrokken.



Figuur 52 Modelleren van entiteiten en relaties

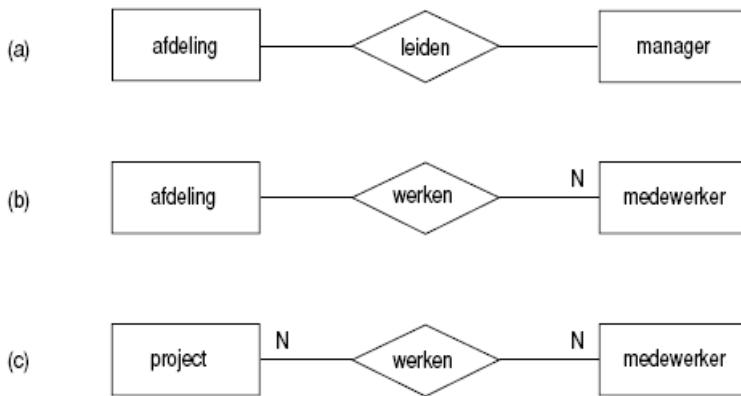
Aard van relatietypen

In zijn oorspronkelijke model beschrijft Chen de aard van het relatietype. Uitgaande van een binair relatietype tussen de entiteitstypen A en B:

- 1:1 – één occurrence van A is gerelateerd aan nul of één occurrence van B en vice versa;
- 1:N – één occurrence van A is gerelateerd aan nul, één of meer occurrences van B, maar elke occurrence van B is gerelateerd aan nul of één occurrence van A;
- N:N – één occurrence van A is gerelateerd aan nul, één of meer occurrences van B, vice versa.

Figuur 53 illustreert de aard van relatietypen en toont de volgende situaties:

- a. Een afdeling wordt geleid door één manager.
Een manager leidt één afdeling.
- b. Een afdeling geeft werk aan meer werknemers (dat wil zeggen: nul, één of meer).
Een werknemer werkt voor één afdeling.
- c. Een werknemer werkt aan meer projecten.
Aan een project werken meer werknemers.



Figuur 53 Aard van relatietypen

6.2 De ISO-variant

De ISO-variant is gedefinieerd door de werkgroep TC97/SC5/WG3 van de International Organization for Standardization (ISO). Ofschoon de werkgroep het bestaan van verschillende varianten heeft onderkend, presenteert ze toch een grafische notatiewijze en een taalsyntaxis voor het beschrijven van het model [ISO]. De volgende concepten zijn karakteristiek voor de ISO-variant:

- een grafische en een tekstuele component;
- de cardinaliteiten van de relatietypen.

Deze concepten worden hieronder verder verklaard.

Grafische en tekstuele component

Volgens de ISO-variant bestaat een ER-model uit een grafische en een tekstuele component. Het grafische gedeelte toont de entiteitstypen en de relatietypen met de bijbehorende cardinaliteiten. Het tekstuele gedeelte beschrijft wat al in het diagram is weergegeven en geeft bovendien beschrijvingen van attribuuttypen en identifiers. Dit gedeelte van de beschrijving is hieronder weergegeven.

Entity-type:	<naam>
Identifier:	<naam>
Description:	<namen>
Relationship-type:	<naam>
Dimension:	<aantal>
Collection:	<namen>
Cardinality:	<namen (min,max)>

De beschrijving van een entiteitstype bestaat uit:

- de naam van het entiteitstype;
- het identificerende attribuuttype of een combinatie van attribuuttypen;
- een opsomming van zijn eigen attribuuttypen.

De beschrijving van een relatietype bestaat uit:

- de naam van het relatietype;
- de dimensie, het aantal entiteitstypen dat met dit relatietype is verbonden;
- de collectie, de namen van de entiteitstypen die met deze associatie zijn verbonden;
- de cardinaliteit per participierend entiteitstype.

De identificerende attribuuttypen en de opsomming van de attribuuttypen zijn complementair aan het diagram. De beschrijvingen van de relatietypen zijn toegevoegd om het tekstuele model compleet te maken.

Cardinaliteiten van relatietypen

Binnen de ISO-variant wordt het concept van functionaliteit gecombineerd met het concept van totaliteit/partialiteit:

- De mogelijke soorten functionaliteit voor een binair relatietype tussen de entiteitstypen A and B zijn:
1:1
1:N
N:N
- Een *totaal* relatietype gedefinieerd tussen de entiteitstypen A en B eist dat iedere occurrence van A en iedere occurrence van B moeten participeren in een occurrence van het relatietype.
Een *partieel* relatietype gedefinieerd tussen entiteitstypen A en B staat toe dat één occurrence van A en één occurrence van B participeren in een occurrence van het relatietype.

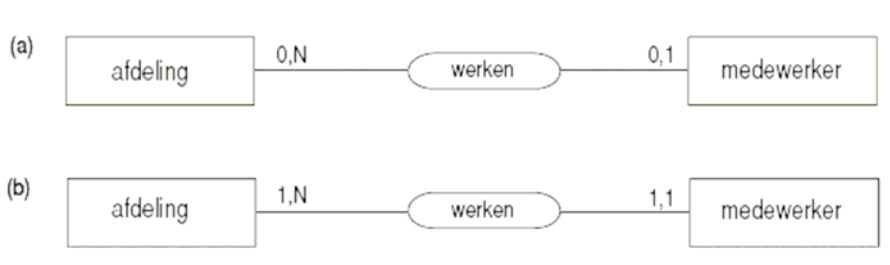
De combinatie van functionaliteit en totaliteit/partialiteit komt tot uiting in de cardinaliteiten van relatietypen. Deze wordt uitgedrukt in een minimum- en maximum-cardinaliteit. Figuur 54 toont de grafische notatie hiervan (de plaatsing is anders dan bij Chen's oorspronkelijke model).

Figuur 54(a) toont dat:

- een afdeling nul of meer medewerkers in dienst heeft;
- een medewerker werkt voor nul of één afdeling.

Figuur 54(b) toont dat:

- een afdeling één of meer medewerkers in dienst heeft;
- een medewerker werkt voor één afdeling.

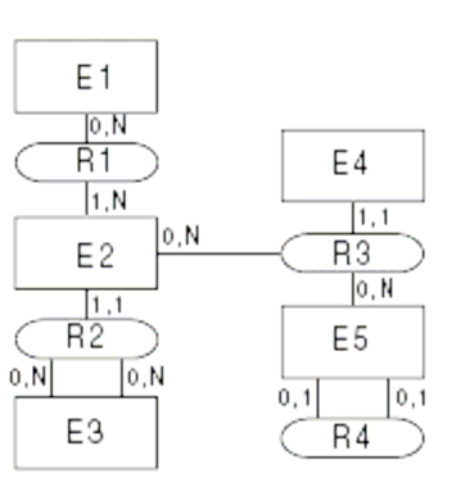


Figuur 54 Concept van cardinaliteiten volgens de ISO-variant

Modelleringsstechniek

Figuur 55 toont een voorbeeld van de mogelijke modelleringsstechnieken volgens de ISO-variant. Dit ER-model schetst de volgende situaties:

- Een binair relatietype R1 tussen de entiteitstypen E1 en E2.
- Een recursief ('involved') relatietype R2, dat tevens is geobjectiveerd. Het recursieve relatietype betreft het entiteitstype E3. De objectivering resulteert in entiteitstype E2. De identificerende attribuuttypen van entiteitstype E2 bestaan uit de identificerende attribuuttypen van de in het relatietype participerende entiteitstypen E3 en E3. De identifier van entiteitstype E2 is: 'I3, I3'.
- Een ternair relatietype R3 tussen de entiteitstypen E2, E4 en E5. De identificerende attribuuttypen van entiteitstype E4 zijn overgeërfd van de entiteitstypen E2 en E5. De identifier van entiteitstype E4 is: 'I3, I3, I5'.
- Een recursief relatietype R4 op entiteitstype E5.



Figuur 55 Modelleringsstechniek volgens de ISO-variant

Entity type: E1	Entity type: E2	
Identifier: I1	Identifier: I3, I3	
Description: I1, A1	Description: A2	
Entity type: E3	Entity type: E4	Entity type: E5
Identifier: I3	Identifier: I3, I3, I5	Identifier: I5
Description: I3, A3	Description: A4	Description: I5, A5

Relationship type: R1	Relationship type: R2
Dimension: 2	Dimension: 3
Collection: E1, E2	Collection: E2, E3,
Cardinality: E1 0,N	Cardinality: E2 1,1
E2 1,N	E3 0,N
	E3 0,N
Relationship type: R3	Relationship type: R4
Dimension: 3	Dimension: 2
Collection: E2, E4, E5	Collection: E5
Cardinality: E2 0,N	Cardinality: E5 0,1
E4 1,1	E5 0,1
E5 0,N	

Bij ‘Dimension’ staat het aantal entiteitstypen (van hetzelfde of verschillende type) vermeld dat participeert in het relatietype. Bij ‘Collection’ staat het aantal verschillende entiteitstypen, dat participeert in het relatietype.

6.3 Ter aanvulling

Praktische aanbeveling

Het is voldoende om in het tekstuele gedeelte te volstaan met de entiteitstypedefinities. De entiteitstypedefinitie bevat naast de naam van het entiteitstype ook de identificerende attribuuttypen en een attribuuttypenlijst.

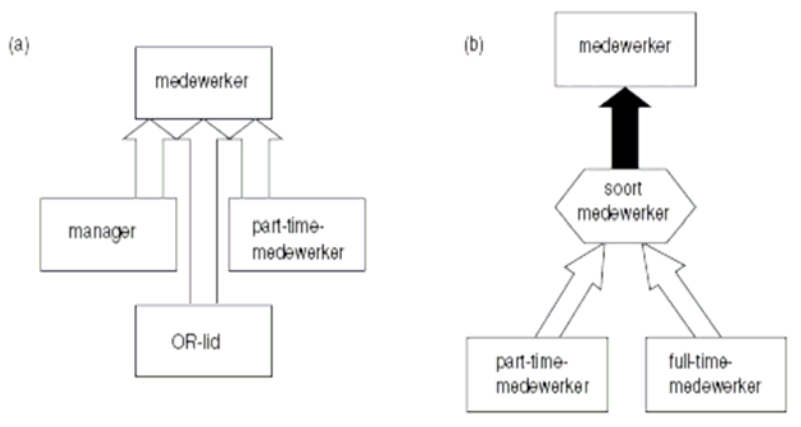
De beschrijvingen van de relatietypen zijn optioneel. De diagrammen, voorzien van cardinaliteitenparen, zijn voldoende.

Subtypen

De ISO-variant voorziet niet in een grafische syntaxis voor subtypen.

Een subtype wordt gedefinieerd wanneer een groep attributen alleen van toepassing is op een subset van de entiteiten van een bepaald type. Binnen dit concept geldt dat entiteiten van het subtype ook behoren tot het supertype, maar niet andersom.

Figuur 56 illustreert het concept ‘subtype’.



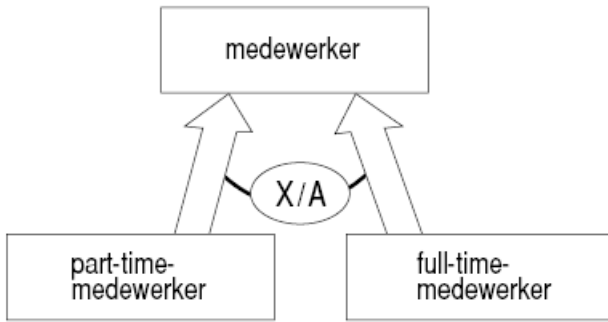
Figuur 56 Subtypen

In figuur 56(a) heeft een medewerker de attributen salarisnummer, naam, adres en salaris. Sommige medewerkers, namelijk zij die manager zijn, hebben tevens het attribuut tekenbevoegdheid. In dit voorbeeld is iedere manager ook een medewerker en heeft daardoor ook de attributen van medewerker. Niet elke medewerker is een manager. Hetzelfde geldt voor een OR-lid. Van enkele medewerkers, zij die in de OR zitten, wil men weten welke functie de medewerker daarin bekleedt (voorzitter, secretaris of lid) en wat de datum van toetreding is.

Daarnaast werken sommige medewerkers part-time. Van hen wil men het part-time-percentage weten. Er zijn dus ook medewerkers die geen manager, OR-lid of parttimer zijn. De subtypen zijn dus niet totaal. De subklassen Manager, OR-lid en Part-time-medewerker zijn niet disjunct: een part-time-manager die lid van de OR is, wordt niet uitgesloten.

In figuur 56(b) zijn de subklassen wel disjunct (exclusief). In dit voorbeeld is een medewerker òf een full-time-medewerker, òf een part-time-medewerker. Het is niet toegestaan dat een medewerker tot meer dan één subklasse behoort. Iedere medewerker behoort tot slechts één subklasse. Dit is in figuur 56(b) grafisch weergegeven door de dichte pijl.

Figuur 57 toont een alternatieve notatie waarin de totaliteits- en exclusiviteitsregels op een NIAM-achtige wijze zijn aangegeven. In dit voorbeeld is iedere medewerker een part-time-medewerker of een full-time-medewerker. Een medewerker behoort tot tenminste één van de subtypen. Dit wordt aangeduid met een 'A'; de totaliteit. Een medewerker kan echter niet zowel een part-time- als een full-timedewerker zijn. Oftewel, de subtypen sluiten elkaar uit. Dit wordt aangegeven met een 'X'; de exclusiviteit. Deze notatiewijze blijkt in de praktijk goed bruikbaar.



Figuur 57 'Praktische' notatiewijze voor subtypen

Informatieregels

De mogelijkheden voor het modelleren van de informatieregels zijn binnen de concepten van de ER-methode beperkt. NIAM en Infomod bieden hiervoor meer mogelijkheden.

De identificerende attribuuttypen – de attribuuttypen die uniek zijn voor een entiteitstype – worden genoteerd in de Identifier-clausule van de beschrijving van het entiteitstype.

Entity type:	MEDEWERKER
Identifier:	Salarisnummer
Description:	Salarisnummer Naam Adres Salaris

Het aangeven van verplichte attribuuttypen – de attribuuttypen die bekend moeten zijn wanneer het entiteitstype bekend is – is alleen mogelijk voor relatietypen. Deze worden gemodelleerd door de minimumcardinaliteit (1).

Bestaansafhankelijkheid tussen entiteitstypen wordt gemodelleerd door het overerven van de identificerende attribuuttype(n) van het entiteitstype waar het 'ondergeschikte' entiteitstype van afhankelijk is.

Entity type:	MEDEWERKER	Entity type:	KIND
Identifier:	Salarisnummer	Identifier:	Salarisnummer
Description:	Salarisnummer Naam Adres Salaris	Description:	Kindnaam

Afhankelijkheidsregels, validatieregels, afleidingsregels en transitierregels worden niet door de ER-methode afgedekt. Deze regels moeten aanvullend worden gespecificeerd. De appendices A en B bevatten hier voorbeelden van.

Submodellen

De ER-methode maakt, anders dan bij Infomod en NIAM, een fundamenteel onderscheid tussen associaties van primaire entiteiten met andere primaire entiteiten (relaties) en associaties van primaire entiteiten en niet-primaire entiteiten (attributen). Dit onderscheid vraagt speciale aandacht bij het integreren van submodellen.

Het volgende voorbeeld illustreert dit. In het onderstaande model komen twee ‘views’ tot uiting.

Entity type: MEDEWERKER	Entity type: AFDELING
Identifier: Salarisnummer	Identifier: Afdelingnummer
Description: Salarisnummer	Description: Afdelingnummer
Naam	Afdelingnaam
Adres	Afdelingadres
Salaris	Budget
Afdelingnaam	

Het linker model toont een entiteitstype Medewerker waarvan men het salarisnummer, naam, adres, salaris en afdelingsnaam wil weten. Het rechter model toont het entiteitstype Afdeling, waarvan afdelingnummer, naam, adres en budget bekend zijn.

Wanneer de twee submodellen worden gecombineerd tot één model, wordt het attribuuotype afdelingnaam bij het entiteitstype Medewerker vervangen door een relatietype tussen Medewerker en Afdeling. Het gecombineerde model ziet er als volgt uit:

Entity type: MEDEWERKER	Entity type: AFDELING
Identifier: Salarisnummer	Identifier: Afdelingnummer
Description: Salarisnummer	Description: Afdelingnummer
Naam	Afdelingnaam
Adres	Afdelingadres
Salaris	Budget
Relationship type: WERKEN	
Dimension: 2	
Collection: Medewerker	
	Afdeling
Cardinality: Medewerker 1,1	
	Afdeling 0,N

7 Naar een relationeel gegevensmodel

7.1 Terminologie van het relationele model

Een relatie of tabel in een relationele database is een tabel van gegevens, waarbij:

- geen twee rijen identiek zijn;
- de volgorde van de rijen niet van belang is;
- iedere kolom een unieke naam heeft;
- de volgorde van de kolommen niet van belang is, alhoewel in sommige implementaties de volgorde betekenis krijgt wanneer de kolommen deel uitmaken van primaire of vreemde sleutels.

Om verwarring met het begrip ‘relatie(type)’ uit de ER-methode te voorkomen spreken we hierna van ‘tabel’ ter aanduiding van het begrip ‘relatie’ uit de relationele theorie.

In de relationele theorie voor relationele databases wordt elke rij een tupel genoemd en elke kolom een veld. Voor de velden kan een lijst van toegestane waarden (domeinen) gelden. Een domein is een verzameling van alle geldige waarden voor een of meer attributen die op dit domein gedefinieerd zijn. Deze waarden behoeven niet noodzakelijkerwijs in een tabel voor te komen.

Elke tabel bevat ten minste één sleutel. Een sleutel is een veld dat of combinatie van velden die tupels uniek kan identificeren. Deze sleutels noemen we kandidaatsleutels. Eén kandidaatsleutel wordt gekozen tot primaire sleutel. De andere worden alternatieve sleutels genoemd.

Een tupel van een tabel kan een referentie hebben naar een tupel van een andere tabel. Deze referentie wordt gelegd via een vreemde sleutel (‘foreign key’). Een vreemde sleutel is een attribuut of een combinatie van attributen in een tabel R_m , waarbij hetzelfde attribuut of dezelfde combinatie van attributen voorkomt als primaire sleutel in een tabel R_n .

Tabellen moeten voldoen aan twee eisen, die van entiteitsintegriteit en referentiële integriteit.

Entiteitsintegriteit betekent dat de waarde van de primaire sleutel nooit NULL (niet aanwezig, leeg) mag zijn. In geval van een samengestelde sleutel betekent dit dat geen van de velden die onderdeel vormen van de sleutel de waarde NULL mag bevatten.

Referentiële integriteit betekent dat op elk moment moet gelden dat alle waarden van de vreemde sleutel in een tabel R_m gelijk moeten zijn aan een waarde van de primaire sleutel in één van de tupels van tabel R_n of de waarde NULL moeten hebben.

Voorbeeld:

Artikelnummer	Omschrijving	Prijs	Voorraad	Leveranciercode
138	Axle	75,00	24	492
286	Journal	1,50	16	822
064	Brush	3,75	96	–
143	Cotter pin	14,25	112	492
092	Bolt	0,25	84	102
185	Nut	0,15	105	102

De tabel Artikel

Leveranciercode	Naam	Plaats
492	Smith & Jones	Tumbleweed
822	Dalton	Frontiertown
102	Denver Grocery	Denver
065	Winston	Dodge City

De tabel Leverancier

Artikelnummer in de tabel Artikel en leveranciercode in de tabel Leverancier zijn primaire sleutels. Leveranciercode in de tabel Artikel is een vreemde sleutel, die refereert naar leveranciercode in de tabel Leverancier.

Artikelnummer en leveranciercode zijn gedefinieerd op hetzelfde domein. Dit domein staat alleen gehele getallen toe van 001 tot en met 999.

De tabellen Artikel en Leverancier volgen de regels van de entiteitsintegriteit: alle tupels hebben een unieke primaire sleutel. De tabellen voldoen ook aan de regels van de referentiële integriteit: alle waarden van de vreemde sleutel leveranciercode in de tabel Artikel verwijzen naar een bestaande primaire sleutel uit de tabel Artikel of hebben de waarde NULL.

Notatiewijze

Doorgaans worden primaire sleutels met een ononderbroken onderstreping en vreemde sleutels met een gestippelde onderstreping aangeduid. In dit boek worden:

- primaire sleutels onderstreept afgedrukt;
- vreemde sleutels <tussen punthaken> geplaatst.

7.2 Naar een relationeel gegevensmodel door normaliseren

Functionele afhankelijkheid

Normaliseren steunt op het (wiskundige) begrip ‘functionele afhankelijkheid’.

Stel, A en B zijn attributen van de tabel R. Dan wordt B functioneel afhankelijk van A binnen R genoemd als bij elk tweetal tupels van R gelijkheid van attribuutwaarde voor A gelijkheid van attribuutwaarde voor B impliceert ($A \rightarrow B$).

Eenvoudig gezegd: bij elke A hoort een bepaalde B.

Voorbeeld: elke medewerker heeft een uniek salarisnummer, één naam en werkt voor één afdeling.

Tabel: Medewerker (salarisnummer, naam, afdeling)

Zowel naam als afdeling zijn functioneel afhankelijk van salarisnummer.

Wat men voor ogen had, moge duidelijk zijn: als een tabel bestaat uit een sleutel en een aantal attributen die daarvan functioneel afhankelijk zijn, dan is de situatie bereikt dat de tabel alleen maar attributen bevat van het entiteittype dat door die tabel wordt afgebeeld en waarvan de entiteiten onderling worden onderscheiden door de sleutel.

Men noemde deze situatie de derde normaalvorm (3NF), het resultaat van drie opeenvolgende, strikt gedefinieerde normalisatiestappen.

Normaliseren

In de eerste stap worden alle repeating groups en afleidbare gegevens (‘procesgegevens’) verwijderd. In de tweede stap wordt ervoor gezorgd dat alle attributen afhankelijk zijn van de volledige sleutel, en niet van slechts een deel ervan. In de derde stap worden ook alle afhankelijkheden tussen attributen onderling verwijderd.

Nu zijn die normaalvormen (1NF, 2NF en 3NF) zeer formeel beschreven. Echter, er bleken nu toch nog situaties voor te komen die wel voldoen aan de letterlijke, formele eisen van 3NF, maar waarbij het toch kan zijn dat een bepaald gegeven op meer dan één plek voorkomt. Toen hebben Boyce en Codd een nieuwe definitie van ‘genormaliseerd zijn’ geformuleerd: de Boyce-Codd-normaalvorm (BCNF). Dit is een definitie die in de praktijk goed voldoet en ook veel gemakkelijker en logischer is dan 3NF.

De theorie van het normaliseren was destijds een grote stap voorwaarts in het vak informatica en heeft een zeer belangrijke aanzet gegeven voor de theorievorming rond informatiemodellering. Bij de huidige praktijk van informatiemodellering zien we normaliseren niet meer als een aparte ontwerpstep, hooguit nog als een controle achteraf.

Normaalvormen

Eerste normaalvorm (1NF)	Een tabel is in de eerste normaalvorm (1NF) als de elementen van de domeinen waarop de attributen van de tabel zijn gedefinieerd ondeelbaar zijn.
Tweede normaalvorm (2NF)	Een tabel is in de tweede normaalvorm (2NF) als ze in 1NF is en alle niet-sleutelattributen volledig functioneel afhankelijk zijn van de kandidaat-sleutel(s).
Derde normaalvorm (3NF)	Een tabel is in de derde normaalvorm (3NF) als ze in 2NF is en alle attributen niet-transitief afhankelijk zijn van de kandidaat-sleutel(s).
BCNF	Een tabel staat in BCNF als ze in 1NF staat en elke determinant een kandidaat-sleutel is voor deze tabel.

Normalisatiestappen

Om een niet-genormaliseerde gegevensverzameling naar de eerste normaalvorm (1NF) te brengen zijn de volgende stappen nodig:

- 1 Het verwijderen van herhalende (groepen van) attributen:
 - 1.1 Inventariseer alle elementaire gegevens.
 - 1.2 Verwijder alle procesgegevens.
 - 1.3 Geef een sleutel aan de tabel.
 - 1.4 Geef een deelverzameling aan die een herhaald aantal malen voorkomt.
 - 1.5 Creëer een nieuwe tabel voor elke zich herhalende deelverzameling; in deze verzameling wordt de sleutel van de oorspronkelijke tabel met de betreffende deelverzameling opgenomen.
 - 1.6 Verwijder de zich herhalende deelverzameling uit de oorspronkelijke tabel.
 - 1.7 Herhaal eventueel de stappen 1.3 tot 1.7.

Om een gegevensverzameling die in 1NF is naar de tweede normaalvorm (2NF) te brengen moeten de volgende stappen worden uitgevoerd:

- 2 Het verwijderen van attributen die functioneel afhankelijk zijn van slechts een gedeelte van de sleutel:
 - 2.1 Bepaal de sleutels (kandidaat-sleutels).
 - 2.2 Beschouw slechts de attributen die niet tot de kandidaat-sleutel(s) behoren.
 - 2.3 Geef de niet-sleutelattributen aan die niet functioneel afhankelijk zijn van de volledige primaire sleutel.
 - 2.4 Formeer een aparte tabel voor ieder deel van de sleutel waar niet-sleutelattributen afhankelijk van zijn.
 - 2.5 Neem in iedere groep de attributen op met het sleuteldeel waarvan ze afhankelijk zijn.

2.6 Verwijder deze attributen uit de oorspronkelijke groep.
Om een gegevensverzameling die in 2NF is naar de derde normaalvorm (3NF) te brengen moeten de volgende (tussen)stappen worden uitgevoerd:

- 3 Het verwijderen van attributen die ook functioneel afhankelijk zijn van andere attributen:
 - 3.1 Geef de attributen aan die ook functioneel afhankelijk zijn van andere attributen.
 - 3.2 Formeer een tabel voor elk attribuut (of elke combinatie van attributen) waarvan andere attributen functioneel afhankelijk zijn. Deze vormen de sleutels van de nieuwe tabellen.
 - 3.3 Neem in elke tabel de hiervan afhankelijke attributen op.
 - 3.4 Verwijder de ‘transitieve’ attributen uit de oorspronkelijke tabel(len).

Boyce-Codd-normalvorm (BCNF)

Een tabel staat in BCNF wanneer ze in 1NF is en iedere determinant een kandidaat-sleutel is voor deze tabel. Een determinant is een attribuut of combinatie van attributen waar een ander attribuut afhankelijk van is. Anders gezegd: de tabel beschrijft slechts één entiteitstype.

Om na te gaan of een gegevensverzameling in BCNF staat kunnen de volgende stappen worden uitgevoerd:

- 4.1 Bepaal de determinanten van de tabel.
- 4.2 Bepaal van elke determinant of hij een kandidaat-sleutel is voor de tabel.

Normaliseren – een voorbeeld

Gegeven de volgende tabel:

Order (ordernummer, orderdatum, orderbedrag, klantnummer, klantnaam, klantadres, klantpostcode, artikelnummer1, prijs1, aantal1, bedrag1, ..., ..., ..., ..., artikelnummerN, prijsN, aantalN, bedragN)

Verder is gegeven:

bedrag = aantal * prijs
orderbedrag = SOM {bedrag1...bedragN}

De primaire sleutel is onderstreept.

De tabel voldoet niet aan de eisen van 1NF:

- er staan afleidbare gegevens (‘procesgegevens’) in: bedrag en orderbedrag

- er staan herhaalde groepen ('repeating groups') in: artikelnummer1 enzovoort.

Om de tabel in 1NF te krijgen worden de afleidbare gegevens verwijderd en de herhaalde groepen in een aparte tabel ondergebracht, die door middel van een 'foreign key' aan de eerste wordt gerelateerd:

Order (ordernummer, orderdatum, klantnummer, klantnaam, klantadres,
 klantpostcode)
Regel (<ordernummer>, artikelnummer, prijs, aantal)

Order voldoet aan de eisen van 2NF. Regel voldoet hier niet aan:

- Prijs is niet afhankelijk van de gehele sleutel, doch slechts van artikelnummer.

Om Regel in 2NF te krijgen moet de 'artikel tabel' afgesplitst worden:

Regel (<ordernummer>, <artikelnummer>, aantal)
Artikel (artikelnummer, prijs)

We hebben nu dus al drie tabellen: Order, Regel en Artikel.

Regel en Artikel voldoen aan de eisen van 3NF (ook BCNF, 4NF, 5NF). Die zijn dus nu 'volledig genormaliseerd'. Met Order ligt dat anders. Intuïtief is dat ook wel duidelijk: er staan 'klantattributen' in de ordertabel.

Order voldoet niet aan de eisen van 3NF:

- Klantnaam, klantadres en klantpostcode zijn transitief (via klantnummer) afhankelijk van ordernummer.

We splitsen de 'klant tabel' dus af:

Order (ordernummer, orderdatum, <klantnummer>)
Klant (klantnummer, klantnaam, klantadres, klantpostcode)

Neem aan dat behalve klantnummer ook de combinaties klantnaam/klantadres en klantnaam/klantpostcode uniek zijn. Dan zijn alle attributen onderdeel van kandidaat-sleutels. Daarmee voldoet de tabel per definitie aan de eisen van 3NF, die immers, net zoals die voor 2NF, voor zogenaamde niet-sleutelattributen zijn gedefinieerd. Dus is aan alle eisen voldaan. Toch is er nog sprake van een afhankelijkheid: klantpostcode is afhankelijk van klantadres.

Aan de eisen van BCNF wordt niet voldaan:

- Klantadres is determinant, maar geen kandidaat-sleutel voor deze tabel.

De postcodetabel wordt afgescheiden:

Klant (klantnummer, klantnaam, <klantadres>)
 Adres (klantadres, klantpostcode)

Het is niet ondenkbaar dat deze laatste normalisatiestap in de praktijk soms, bewust of onbewust, achterwege blijft. Zoals in dit voorbeeld: eigenlijk moet een apart ‘postcodeboek’ worden gehanteerd, maar de postcode zal toch vaak in de klanttabel worden opgenomen.

Soms ook wordt, om performanceredenen voor retrieval, een tabel bewust gedenormaliseerd’ (van 3NF naar 2NF). Bijvoorbeeld: klantgegevens staan, behalve in de klanttabel, ook in de ordertabel. DBMS’ en en generatoren bieden vaak de mogelijkheid dat de inhoudelijke correspondentie van de waarden door middel van regels gewaarborgd wordt.

Het volledige relationele model in BCNF ziet er als volgt uit:

Klant (klantnummer, klantnaam, <klantadres>)
 Adres (klantadres, klantpostcode)
 Artikel (artikelnummer, prijs)
 Order (ordernummer, orderdatum, <klantnummer>)
 Regel (<ordernummer>, <artikelnummer>, aantal)

7.3 Naar een relationeel gegevensmodel door informatiemodellering

Zoals we al eerder aangaven, zien wij normaliseren meer als een controle achteraf dan als een specifieke ontwerpactiviteit. De weg die wij binnen MAD voorstaan is eerst informatie modelleren en dan vanuit het informatiemodel omzetten naar een relationeel gegevensmodel. Hieronder volgen enkele handvatten voor deze transformatie.

Entiteitstypen en attribuuttypen

Elk entiteitstype uit het informatiemodel wordt in het gegevensmodel gerepresenteerd door een tabel. Dit geldt ook voor de entiteitstypen die een objectivering zijn van een M:M-relatietype. Ook de zogenaamde ‘zwakke’ entiteitstypen worden als een aparte tabel gerepresenteerd. Een zwak entiteitstype is een entiteitstype waarvan het bestaan afhankelijk is van een ander entiteitstype. In het onderstaande voorbeeld is het bestaan van het entiteitstype Kind afhankelijk van het bestaan van het entiteitstype Medewerker. Binnen het gezichtspunt van het bedrijf is het Kind alleen bekend wanneer ook de Medewerker bekend is.

Medewerker (personeelsnummer, naam, adres, woonplaats)
 Kind van medewerker (<personeelsnummer>, kindnaam)

Elk attribuuttype uit het informatiemodel wordt in het gegevensmodel gerepresenteerd door een veld (kolom). Voor meerwaardige attribuuttypen wordt een extra tabel gecreëerd. Voorbeeld: een medewerker spreekt meer talen.

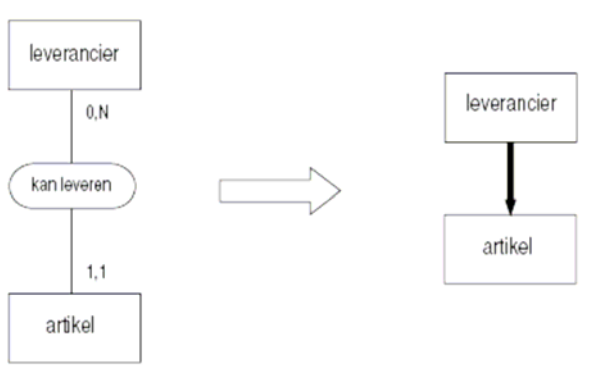
Medewerker (personeelsnummer, naam, adres, woonplaats)
 Taal-Medewerker (<personeelsnummer>, taal)

Relatietypen

Vervolgens worden de relatietypen uit het informatiemodel op het relationele model afgebeeld. Afhankelijk van de cardinaliteiten wordt de relatie gerepresenteerd door vreemde sleutels ('foreign keys') of door een aparte tabel. In het navolgende behandelen we aan de hand van enkele voorbeelden de volgende relatietypen:

- 1:N;
- N:N;
- ternair (objectivering);
- N-air;
- recursief 1:N;
- recursief N:N;
- geobjectiveerd recursief N:N.

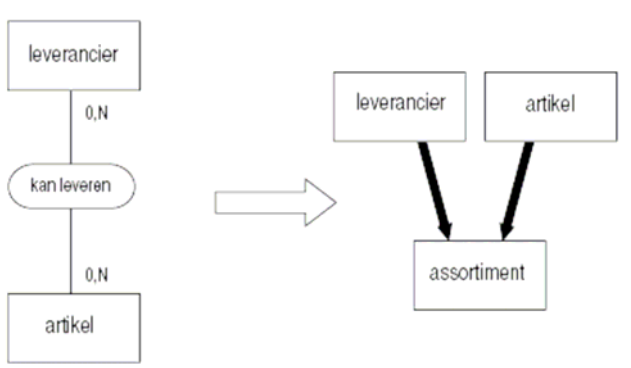
Een 1:N-relatietype (figuur 59) wordt in het gegevensmodel gerepresenteerd door een vreemde sleutel te definiëren in de tabel die het entiteittype representeert dat meerdere keren kan voorkomen. Deze vreemde sleutel verwijst naar de primaire sleutel van de tabel die het entiteittype representeert dat één keer voorkomt.



Figuur 58 1:N-relatietype

Leverancier (leveranciercode, naam, adres, plaats)
 Artikel (artikelnummer, omschrijving, <leveranciercode>)

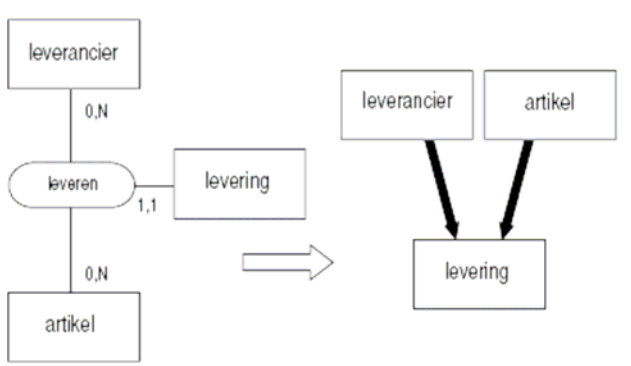
Een N:N-relatietype (figuur 59) resulteert in een extra tabel, die het relatietype representeert. De tabel bestaat ten minste uit de twee primaire sleutels van de twee tabellen die de geassocieerde entiteitstypen representeren.



Figuur 59 N:N-relatietype

Leverancier (leveranciercode, naam, adres, plaats)
 Artikel (artikelnummer, omschrijving)
 Assortiment (<leveranciercode>, <artikelnummer>)

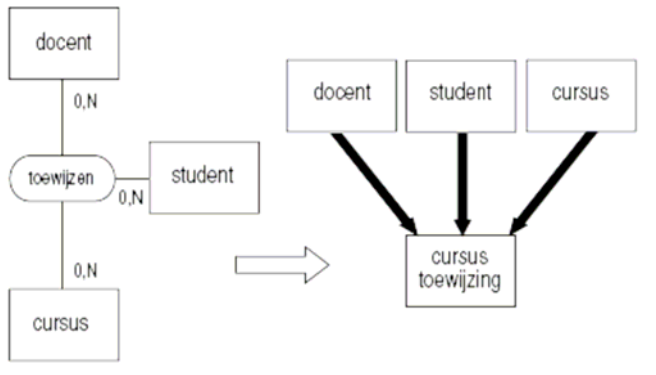
Een *ternair* relatietype (figuur 60), waarbij één entiteittype de cardinaliteit 1,1 heeft (meestal het geobjectiveerde entiteittype), wordt gerepresenteerd door de tabel die het geobjectiveerde entiteittype representeert. Deze tabel heeft twee vreemde sleutels, elk refererend naar de primaire sleutel van één van de andere tabellen.



Figuur 60 Geobjectiveerd ternair relatietype

Leverancier (leveranciercode, naam, adres, plaats)
 Artikel (artikelnummer, omschrijving)
 Levering (<leveranciercode>, <artikelnummer>, datum, hoeveelheid)

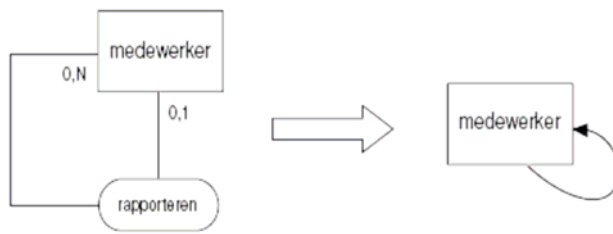
Een *N-air* relatietype (figuur 61), waarbij alle entiteittypen N keer kunnen voorkomen, resulteert in een extra tabel, die het relatietype representeert. De tabel bestaat uit ten minste de drie primaire sleutels van de drie tabellen die de geassocieerde entiteittypen representeren.



Figuur 61 N-air relatietype

Docent (docent-id, naam, adres)
 Student (studentnummer, naam, adres)
 Cursus (cursuscode, onderwerp, cursustype)
 Toewijzing (<docent-id>, <studentnummer>, <cursuscode>)

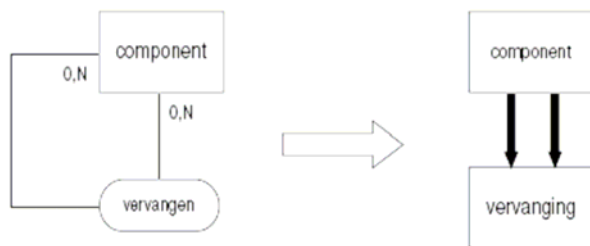
Een *recursieve 1:N-relatietype* (figuur 62) wordt gerepresenteerd door een vreemde sleutel te definiëren in de tabel die het entiteitstype representeert. Deze vreemde sleutel verwijst naar de primaire sleutel van dezelfde tabel.



Figuur 62 Recursief 1:N-relatietype

Medewerker (salarisnummer, naam, adres, <rapporteert-aan-salarisnummer>)

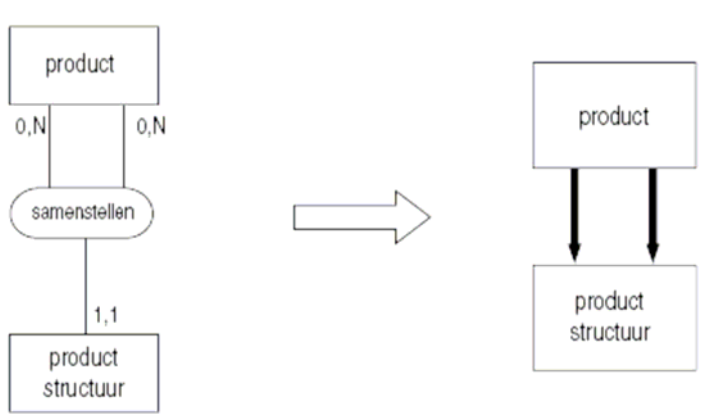
Een *recursieve N:N-relatietype* (figuur 63) resulteert in een extra tabel die de relatie representeert. De tabel bestaat ten minste uit een primaire sleutel bestaande uit twee keer de primaire sleutel van de tabel die het geassocieerde entiteitstype representeert.



Figuur 63 Recursief N:N-relatietype

Component (componentcode, omschrijving, elektrische en fysische eigenschappen)
 Vervanging (<vervangende_componentcode>, <vervangen_componentcode>)

Een *geobjectiveerd recursieve N:N-relatietype* (figuur 64), waarbij het geobjectiveerde entiteitstype de cardinaliteit 1,1 heeft, wordt gerepresenteerd door de tabel die het geobjectiveerde entiteitstype representeert. Deze tabel heeft twee vreemde sleutels, elk refererend naar de primaire sleutel van de tabel die het andere entiteitstype representeert.



Figuur 64 Geobjectiveerd recursief N:N-relatietype

Product (productnummer, omschrijving)
 Product-structuur (<parent_productnummer>, <child_productnummer>, aantal)

Appendix A – Casus: de boekenclub

A.1 Het bedrijfsactiviteitenmodel

Maak aan de hand van de onderstaande beschrijving een bedrijfsactiviteitenmodel voor de boekenclub.

Algemene beschrijving

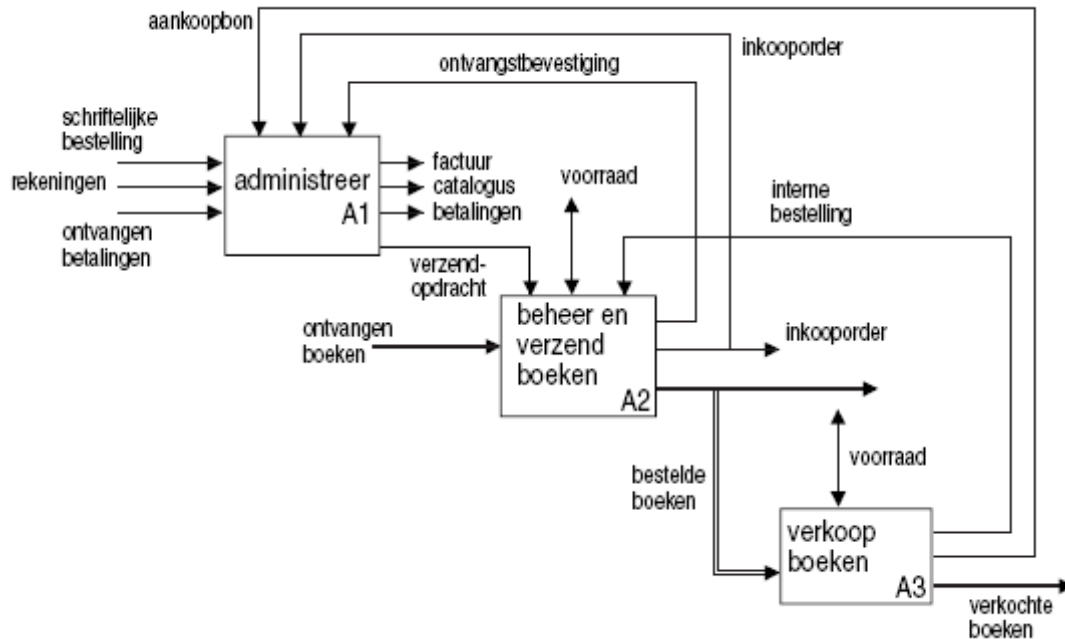
De boekenclub bestaat uit een administratie, een magazijn en een clubwinkel. De verkoop van boeken geschiedt uitsluitend aan clubleden. Dat kan op twee manieren. Leden kunnen een schriftelijke bestelling doen uit de catalogus, of ze kunnen zichervoegen bij de clubwinkel, waar de boeken direct meegenomen kunnen worden.

De administratie registreert de bestellingen en stuurt verzendopdrachten naar het magazijn. De administratie stelt ook eens per drie maanden de catalogus samen, die naar alle leden wordt verzonden, samen met de factuur. Deze factuur brengt de bestelde en/of de bij de clubwinkel afgehaalde boeken van het voorbije kwartaal in rekening. Bij aanvaarding van hun lidmaatschap hebben de clubleden zich verplicht het totale factuurbedrag binnen drie weken na dagtekening te voldoen. De administratie registreert de binnengekomen betalingen.

Het magazijn verzendt op basis van de verzendopdrachten van de administratie de bestelde boeken naar de leden. De winkel wordt vanuit het magazijn beleverd. Het magazijn is zelf verantwoordelijk voor het op peil houden van de voorraad en plaatst van tijd tot tijd bestellingen (inkooporders) bij de uitgevers. Een afdruk hiervan wordt naar de administratie gestuurd. Boeken die besteld zijn bij de uitgever worden in het magazijn in ontvangst genomen. Het magazijn stuurt een ontvangstbevestiging naar de administratie, die op haar beurt de rekeningen van de uitgevers voldoet.

In de clubwinkel kunnen de leden de boeken inkijken en uitkiezen. Als zij een boek aanschaffen, wordt een kopie van de aankoopbon verstuurd naar de administratie. De beheerder van de winkel is zelf verantwoordelijk voor de voorraad in de winkel. Daartoe plaatst hij interne bestellingen bij het magazijn.

Uitwerking



Figuur 73 Diagram A0: Boekenclub

Toelichting

Bij het modelleren van bedrijfsactiviteiten zijn verschillende oplossingen mogelijk. Het is echter geen kwestie van ‘goed’ of ‘fout’. Meestal is er voor elke oplossing wel iets te zeggen. Toch is er in het bijzonder een aantal criteria waarop moet worden gelet:

- de herkenbaarheid van het model naar de gebruikersorganisatie;
- het in balans zijn van de activiteiten;
- het gezichtspunt van waaruit wordt gemodelleerd;
- het doel van het model.

Met het oog op de herkenbaarheid gaat onze voorkeur uit naar een oplossing met drie activiteiten: ‘Administreer’, ‘Verzend boeken’ en ‘Verkoop boeken’. Deze indeling komt één op één overeen met de organisatorische indeling in administratie, magazijn en winkel. Eventueel kan de activiteit ‘Administreer’ verder worden ontleed in ‘Factureer’, ‘Doe en ontvang betalingen’ en ‘Verzend catalogus’. Mogelijk dat deze decompositie leidt tot ontkoppelde activiteiten.

Het gebalanceerd zijn van de activiteiten (‘horizontal balancing’) wordt geweld aangedaan wanneer een oplossing wordt gekozen met zes of meer activiteiten: ‘Factureer’, ‘Doe en ontvang betalingen’, ‘Verzend catalogus’, ‘Verzend boeken’ en ‘Verkoop boeken’. De laatste twee activiteiten zijn qua gewicht (omvang) zwaarder (groter) dan de overige. Ook voor wat betreft dit aspect gaat de voorkeur uit naar de oplossing met drie activiteiten.

Tot slot kan het doel van het modelleren en het gezichtspunt van waaruit dit wordt uitgevoerd ook nog van invloed zijn op de groepering van activiteiten. Wanneer het doel is inzicht te verschaffen in het huidige bedrijfsproces om te komen tot een inventarisatie van activiteiten die voor automatisering in aanmerking komen, is een opdeling in drie activiteiten werkbaar. Het gezichtspunt van waaruit dit wordt gedaan is dan bijvoorbeeld dat van de algemeen directeur. Wanneer het doel is het in kaart brengen van het logistieke proces en het gezichtspunt is dat van de logistiek manager, dan is een oplossing van vijf of zes activiteiten, waarbij de nadruk ligt op de fysieke goederenstroom, wellicht zinvoller.

Bij de naamgeving van de activiteiten zijn er twee aspecten waarmee men rekening moet houden:

- De vlag moet de lading dekken.
- De nadruk moet liggen op het producerende karakter in plaats van op het ontvangende of verwerkende karakter.

Zo is 'Administreer' beter dan 'Factureer' en is 'Verzend boeken' beter dan 'Verwerk verzendopdracht'. Overigens is 'Administreer' ook nog wat te mager, gelet op de uitgaande informatiestroom 'catalogus'.

Ook in de naamgeving voor informatiestromen moet men zorgvuldig zijn. Het kan niet zo zijn dat 'betalingen' een activiteit ingaan en dat 'betalingen' die zelfde activiteit uitgaan. Door een goede naamgeving kan men het onderscheid aanduiden: 'ontvangen betalingen' en 'betalingen'.

Overigens is het altijd raadzaam om zowel voor de activiteiten als voor de informatiestromen een korte omschrijving op te stellen. Deze omschrijving, twee of drie regels, bevat een nadere definitie van de activiteit dan wel de informatiestroom.

Het onderscheid tussen invoer- en controle-informatiestromen is altijd een punt van discussie. 'Verzendopdracht', 'inkooporder' en 'ontvangstbevestiging' zijn duidelijke voorbeelden van controle. Ze ondergaan geen transformatie, maar zijn 'triggers' voor de activiteiten. In het algemeen wordt gesteld dat controlestromen geen transformatie ondergaan; ze sturen activiteiten aan of zijn benodigd (worden geraadpleegd) door de activiteit voor het produceren van de uitgaande stromen. Een voorbeeld is 'Bereken belastingteruggave'. Hiervoor zijn als controlestromen 'belastingregels' (vastgelegd in de almanak) en 'tarieven' (volgens de schijven) nodig. Beide ondergaan geen transformatie, maar zijn wel als informatie benodigd voor het berekenen van het bedrag van teruggave.

Een apart geval is 'Voorraad'. Deze informatiestroom vervult de rol van een 'trigger' naar aanleiding waarvan een inkooporder of interne bestelling wordt geplaatst, maar ondergaat tegelijkertijd een wijziging (de voorraad wordt gemuteerd). Hierbij is, zowel bij A2 als bij A3, gebruikgemaakt van een verkorte notatie, waarbij het controle-aspect dominant is.

Dit ligt iets lastiger voor de informatiestromen 'schriftelijke bestelling' en 'aankoopbon'. Men kan zich afvragen of beide een transformatie ondergaan. De inhoud van de 'schriftelijke bestelling' ondergaat geen verandering: na verwerking gaat het nog steeds

over dat ene boek voor dat ene lid. Aan de andere kant ondergaat de ‘schriftelijke bestelling’ wellicht een statusverandering van ‘ontvangen’ naar ‘verwerkt’. In het algemeen gaat onze voorkeur bij informatieverwerkende processen uit naar de input. Voor de ‘aankoopbon’ ligt dit weer iets anders. Het betreft hier een kopie, enkel bedoeld om het factureren op te starten: een ‘trigger’ dus en daarmee een duidelijke controlestroom.

Het al dan niet apart tekenen van materiestromen en informatiestromen hangt nauw samen met het doel van modelleren. In het algemeen hoeven we deze twee niet te scheiden. De informatiestroom ‘ontvangen boeken’ bevat een doos met boeken afkomstig van een uitgever, die natuurlijk vergezeld gaat van een pakbon met de daarop relevante informatie. Hetzelfde geldt voor de informatiestroom ‘Bestelde boeken’. Een splitsing wordt pas relevant wanneer de bestemming van beide stromen verschillend is. Een voorbeeld hiervan is ‘Verkocht boek’ (materie + informatie) en ‘Aankoopbon’ (alleen informatie). Materiestromen kunnen als dubbele pijlen worden getekend. Dit geldt ook als we zowel de materie als informatiestroom bedoelen.

A.2 Het informatiemodel

Maak een informatiemodel voor de boekenclub. Vul dit model aan met informatieregels. Indien u zich genoodzaakt voelt bepaalde keuzes of aannames te doen, dient u deze toe te lichten.

Aanvullende informatie

Van een lid registreert men: uniek lidnummer, naam, voorletters, ingangsdatum, einddatum. Voor de adressering kan worden volstaan met postcode en huisnummer. Indien een lid daar geen bezwaar tegen heeft, worden ook de sekse en de geboortedatum geregistreerd. Een nieuw lid moet ten minste vier maanden lid blijven.

Van een boek worden ISBN, titel, prijs, verzendkosten, genre, uitgever en auteur geregistreerd. Een auteur (naam en nationaliteit) kan één of meerdere boeken hebben geschreven. In geval van meerdere auteurs wordt alleen de eerste auteur geregistreerd. Niet elk boek hoeft door een auteur te zijn geschreven (bijvoorbeeld in het geval van een door de uitgever samengestelde bundel). Ten behoeve van het voorraadbeheer heeft elk boek een minimum- en een actuele voorraad.

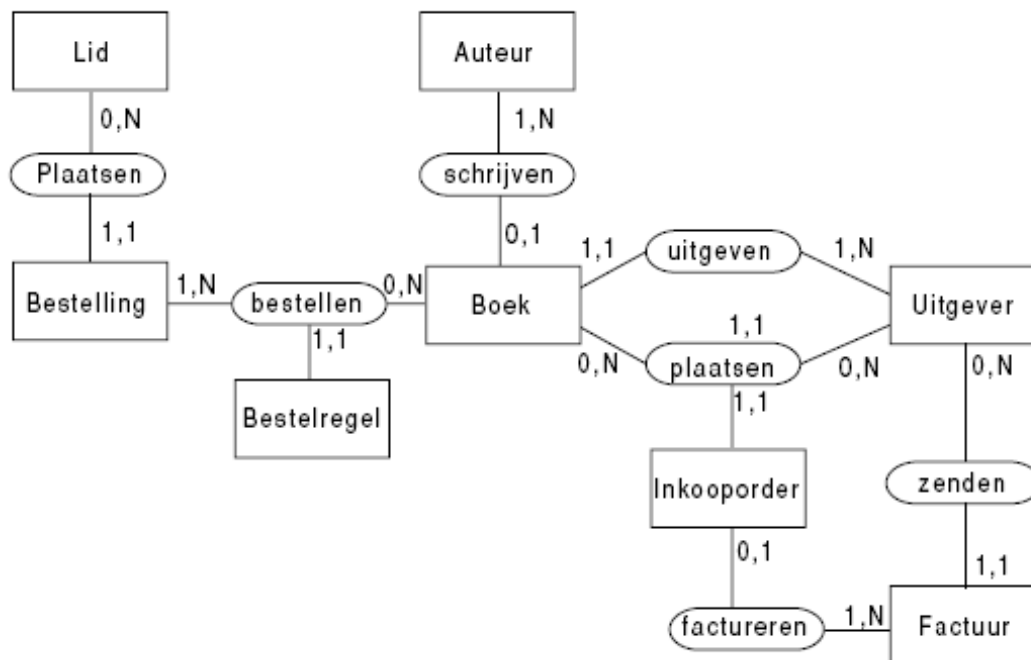
Van een bestelling worden, naast een uniek bestelnummer, de besteldatum, het aantal bestelde exemplaren en het totale bedrag geregistreerd. Tevens houdt men bij op welke datum een besteld boek is verzonden en op welke datum het is betaald. Met behulp van één bestelling kunnen één of meer exemplaren van verschillende boeken tegelijkertijd worden besteld.

Inkooporders hebben een uniek nummer en een datum. Per inkooporder wordt de ontvangstdatum geregistreerd. De facturen die de uitgevers sturen worden ook geadministreerd: het factuurnummer van de uitgever, de datum, de gefactureerde

inkooporders en het totaalbedrag. Tevens wordt de datum geregistreerd waarop de factuur wordt betaald.

Uitgevers hebben een unieke naam en een adres. Van elk boek wordt bijgehouden door welke uitgever het wordt geleverd.

Uitwerking



Figuur 74 Informatiestructuurdiagram

Entiteittypebeschrijvingen

Entity type: Lid

Identifier: lidnummer

Description: lidnummer, naam, voorletters, ingangsdatum, einddatum, postcode, huisnummer, sekse, geboortedatum

Entity type: Boek

Identifier: ISBN

Description: ISBN, titel, prijs, genre, minimumvoorraad, actuele voorraad, verzendkosten

Entity type: Auteur

Identifier: naam

Description: naam, nationaliteit

Entity type: Bestelling

Identificerend: bestelnummer
Beschrijving: bestelnummer, besteldatum, orderbedrag, betaald

Entiteitstype: Bestelregel
Identificerend: bestelnummer, volgnummer
Beschrijving: volgnummer, aantal, bedrag, verzenddatum, betaaldatum

Entiteitstype: Uitgever
Identificerend: naam
Beschrijving: naam, adres

Entiteitstype: Inkooporder
Identificerend: inkoopnummer
Beschrijving: inkoopnummer, datum, aantal, ontvangstdatum

Entiteitstype: Factuur
Identificerend: factuurnummer
Beschrijving: factuurnummer, datum, bedrag

Identificerende attributen

(zie Identificerend)

Verplichte attributen

Lid: lidnummer, naam, voorletters, ingangsdatum, postcode, huisnummer

Boek: boeknummer, titel, prijs, genre

Auteur: naam, nationaliteit

Bestelling: bestelnummer, besteldatum, betaald

Bestelregel: bestelnummer, volgnummer, aantal

En de verplichte relatietypen (minimumcardinaliteit = 1) uit het informatiestructuurdiagram.

Validatieregels

Tussen de eind- en de begindatum van een lid(maatschap) liggen ten minste vier maanden.

Afhankelijkheidsregels

Als een inkooporder voor een bepaald boek bij een uitgever wordt geplaatst, moet al gelden dat die uitgever dat boek uitgeeft.

Er bestaat een afhankelijkheid tussen 'factureren', 'zenden' en 'plaatsen'.

Afleidingsregels

Orderbedrag is de som van de orderregelbedragen van de gerelateerde orderregels.
Orderregelbedrag = (prijs + verzendkosten) * aantal.

Transitieregels

(geen)

A.3 Het specificatiemodel

De boekenclub wil stap voor stap, incrementeel, tot volledige automatisering overgaan. De eerste activiteit die ondersteund dient te worden, is het registreren van bestellingen door leden.

Maak voor dit deel van het boekenclubsysteem een specificatie, bestaande uit een relationeel (grafisch en tekstueel) model met aanvullende regels. Het relationeel model dient alleen die entiteitstypen af te beelden die benodigd zijn voor dit increment. Van die entiteitstypen dienen wel alle attributen en onderlinge relaties in het relationele model gerepresenteerd te worden.

Aanvullende informatie

Van het systeemdeel moet ook managementinformatie beschikbaar zijn, zoals regionale verdeling van de bestellingen (postcode) en bestelde boeken per auteur. Het totaalbedrag van een bestelling moet worden berekend uit de som van de bijbehorende bestelregelbedragen; deze laatste worden berekend volgens: bedrag = (prijs + verzendkosten van het boek) * aantal. Als de actuele voorraad van een boek onder de minimumvoorraad komt, dient dit gesignaleerd te worden.

Uitwerking



Figuur 75 Gegevensstructuur

Lid	(<u>lidnummer</u> , naam, voorletters, ingangsdatum, einddatum, postcode, huisnummer, sekse, geboortedatum)
Boek	(<u>ISBN</u> , titel, prijs, genre, minimumvoorraad, actuele voorraad, verzendkosten, <auteurcode>)
Auteur	(<u>auteurcode</u> , naam, nationaliteit)
Bestelling	(<u>bestelnummer</u> , besteldatum, orderbedrag, postcode, <lidnummer>)
Bestelregel	(< <u>bestelnummer</u> >, <u>volgnummer</u> , aantal, bedrag, verzenddatum, betaaldatum, <ISBN>)

Constraints

Sekse = {M,V}

Event-getriggerde regels

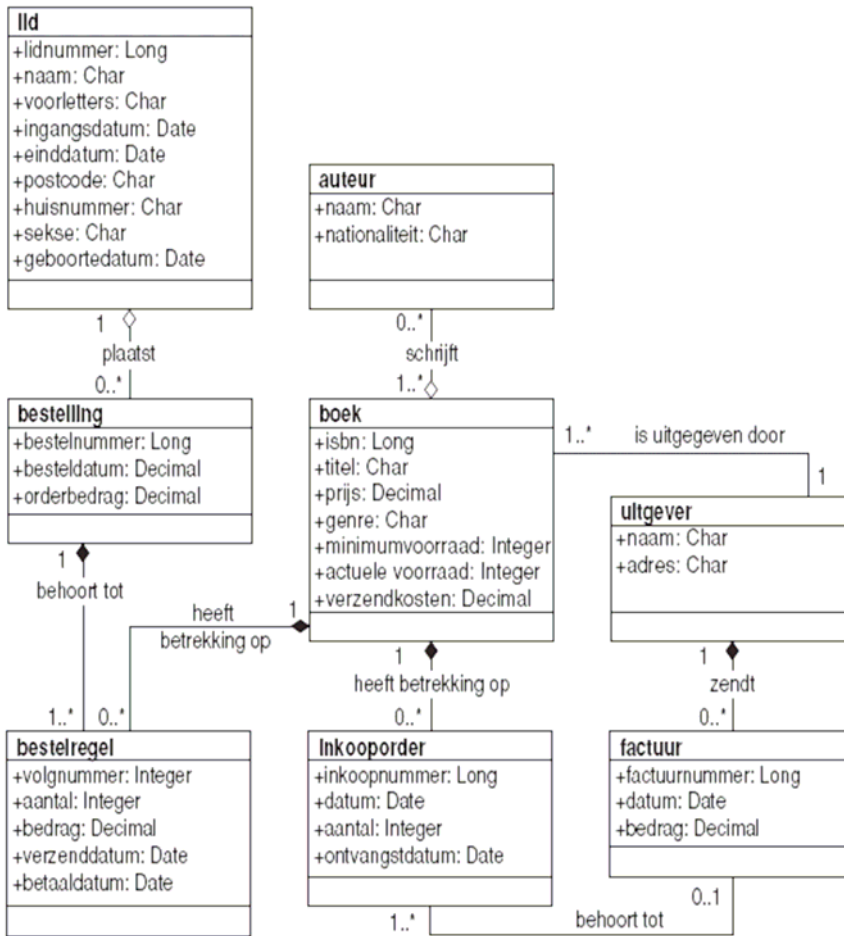
on: POST-CHANGE Lid.einddatum
 if: Lid.begindatum, Lid.einddatum NOT NULL
 then: Lid.einddatum – Lid.begindatum >= 4 maanden

on: PRE-INSERT Bestelling
 if: Bestelling.lidnummer NOT NULL
 then: Bestelling.postcode = Lid.postcode

on: POST-CHANGE Boek.prijs, Bestelregel.aantal
 if: Boek.prijs, Bestelregel.aantal NOT NULL
 then: Bestelregel.bedrag = (Bestelregel.prijs + Boek.verzendkosten) * Bestelregel.aantal
 on: PRE-INSERT, PRE-UPDATE Bestelregel
 if: –
 then: Bestelling.bedrag = SOM (Bestelregel.bedrag)

on: PRE-INSERT, PRE-UPDATE Bestelregel
 if: aantal NOT NULL
 then: Boek.actuele voorraad = Boek.actuele voorraad – Bestelregel.aantal

on: POST-CHANGE Boek.voorraad
 if: voorraad <= minimumvoorraad
 then: Procedure Signaleer ‘Voorraad onder minimum’



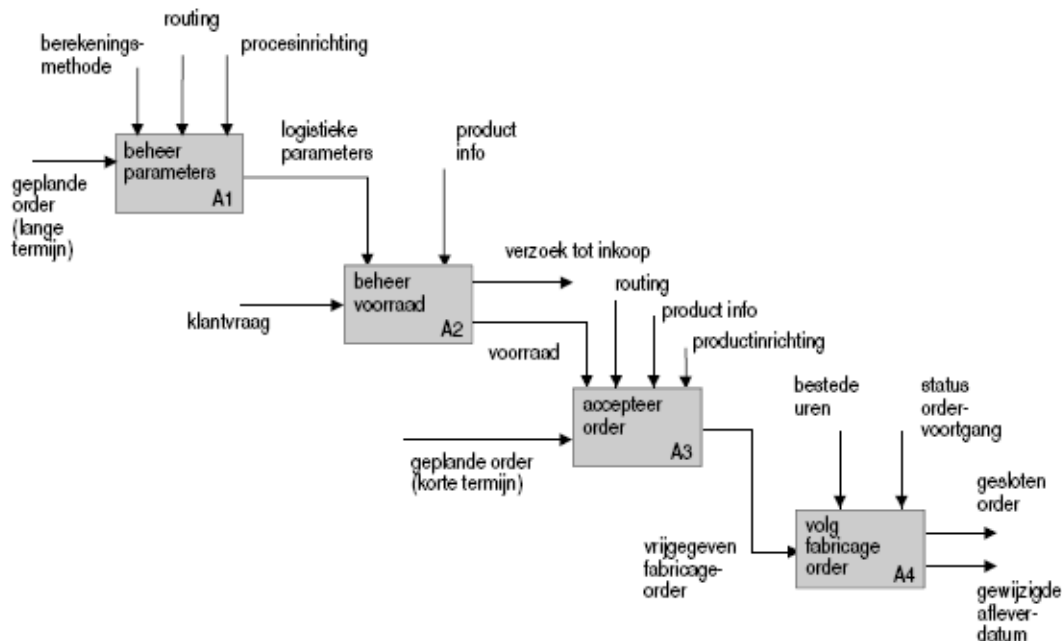
Figuur 76 Gegevensstructuur in UML-notatie (Uitgever, Inkooporder en Factuur zijn toegevoegd)

Appendix B – Casus: de fabriek

B.1 Het bedrijfsactiviteitenmodel

Een fabriek maakt onderdelen voor elektronische apparatuur. Men wil de productiebesturing ondersteunen met een geautomiseerd informatiesysteem. Men wil daarvoor eerst inzicht verkrijgen in het bedrijfsproces van de fabriek. Hiertoe wordt een bedrijfsanalyse uitgevoerd. In een eerste stap wordt gekeken naar de orderacceptatie als onderdeel van de productiebesturing. Hiervoor is reeds in een eerste analyse het bedrijfsproces rond de productiebesturing in kaart gebracht.

Maak met behulp van het onderstaande activiteitendiagram en aan de hand van de beschrijving van de orderacceptatie een decompositie voor activiteit A3 (Accepteer order).



Figuur 77 Diagram A0: Productiebesturing

Orderacceptatie

Enmaal per week komen orders binnen, die ruim van tevoren zijn gepland. Achtereenvolgens wordt gecontroleerd of er voldoende voorraad is om de order te gaan maken, of de fabricage-informatie compleet is en of er voldoende capaciteit in de fabriek aanwezig is om de bestelde producten voor de gevraagde leverdatum te leveren. Wanneer aan deze voorwaarden is voldaan, is de order geaccepteerd en wordt deze vrijgegeven voor fabricage.

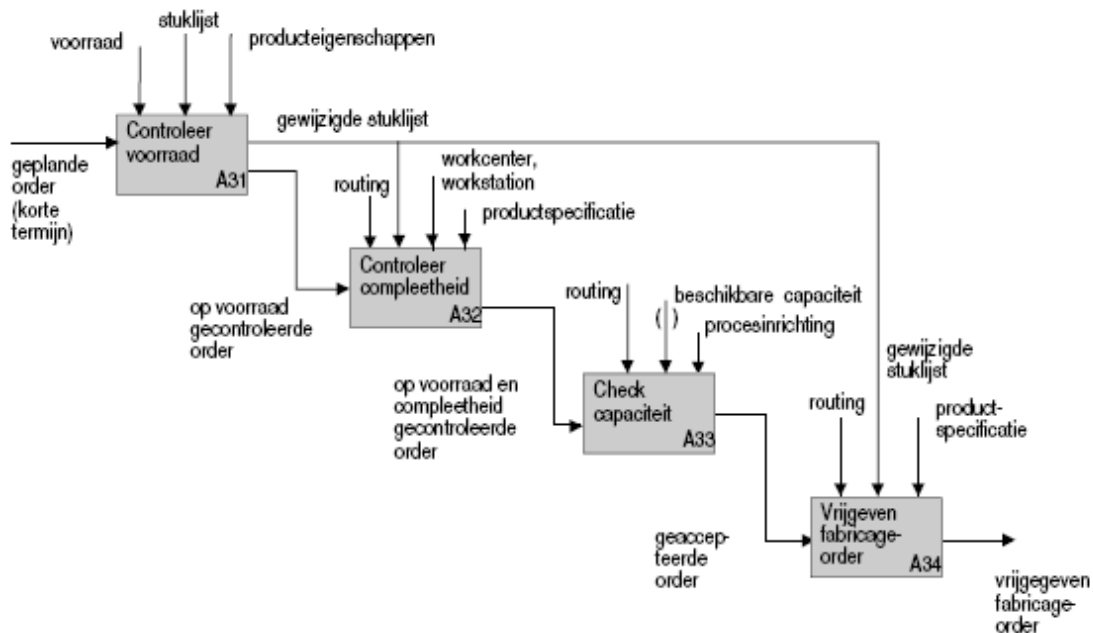
Aan de hand van de stuklijst wordt bepaald welke onderdelen (producten) benodigd zijn. Hiervan wordt gecontroleerd of er voldoende voorraad is. Naar aanleiding van deze controle kan het voorkomen dat een bepaald product in onvoldoende mate aanwezig is. Er wordt dan naar een vervanging van dat product gezocht, met uiteraard dezelfde elektrische en fysieke eigenschappen. Op grond hiervan kan de stuklijst worden aangepast.

De controle van de fabricage-informatie op compleetheid heeft betrekking op de productspecificatie, de routing en de (gewijzigde) stuklijst. Hiervoor is informatie over de procesinrichting (workcenters en workstations) van de fabriek nodig.

Tijdens de globale capaciteitscheck wordt nagegaan of er voldoende capaciteit in de fabriek beschikbaar is om de order voor de gevraagde leverdatum af te handelen. Hiertoe wordt de gevraagde capaciteit berekend en vergeleken met de beschikbare capaciteit. De gevraagde capaciteit wordt berekend uit bewerkingstijden en insteltijden. Hiervoor is informatie over de routing en de procesinrichting nodig.

Bij het vrijgeven van de order wordt de actuele fabricage-informatie (productspecificatie, routing en de eventueel gewijzigde stuklijst) meegezonden. Het geheel, order en fabricage-informatie, wordt in de fabriek ‘fabricage-order’ genoemd. Na vrijgave heeft deze de status ‘vrijgegeven’.

Uitwerking



Figuur 78 Diagram A3: Accepteer order

Producteigenschappen, Stuklijst en Productspecificatie zijn deelstromen van Productinformatie.

B.2 Het informatiemodel

Maak met behulp van het bedrijfsmodel en aan de hand van de onderstaande beschrijving een informatiemodel (informatiestructuur, entiteittypebeschrijvingen en informatieregels).

De informatiebehoefte van de fabriek

Een klant (unieke klantcode, naam, adres, telefoonnummer) van het bedrijf bestelt een product (productnummer, omschrijving). Hierbij wordt het aantal en de gewenste leverdatum genoteerd. Een order heeft altijd betrekking op slechts één product. Orders worden van elkaar onderscheiden door een uniek ordernummer. De benodigde capaciteit om een order af te handelen wordt berekend. De order is van een bepaald type (A, B of C) en kent een aantal statussen. Van een product wordt bijgehouden wanneer het voor het laatst geproduceerd is en wat tot nu toe de totale geproduceerde hoeveelheid is.

Een product kan samengesteld zijn uit andere producten. Een samengesteld product kan bestaan uit meer producten en een bepaald product kan deel uitmaken van meer producten. Van een samenstelling wordt het productnummer van de ‘parent’, het productnummer van de ‘child’ en een prefix vastgelegd. Deze combinatie is tevens uniek voor een samenstelling. Bovendien wordt het aantal componenten van de samenstelling vastgelegd.

Een product kan worden gemaakt volgens een aantal routingen (routingnummer, effectieve startdatum, effectieve einddatum). Een routing geldt slechts voor één product. Een product heeft meer voorgedefinieerde routingen. Voor iedere order wordt een bepaalde routing gekozen. De gekozen routing moet deel uitmaken van de voor dat product voorgedefinieerde routingen.

Routingen worden van elkaar onderscheiden door een uniek routingnummer. Een routing loopt langs een aantal workcenters (workcentercode, naam). Voor iedere routing wordt per workcenter een algemene instructietekst gegeven (bijvoorbeeld een waarschuwing of een raadgeving), evenals de volgorde waarin de workcenters worden afgelopen. Een routing kent een effectieve startdatum en een effectieve einddatum. Buiten deze periode mag de routing niet worden gebruikt.

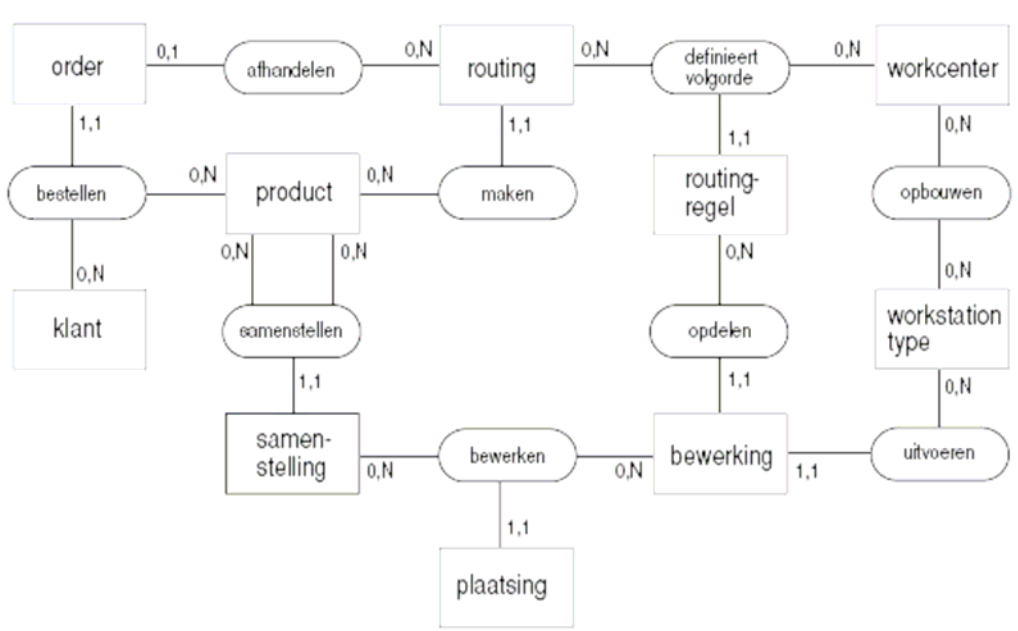
Een routing is per workcenter opgedeeld in bewerkingen. Van een bewerking wordt de volgorde binnen het workcenter vastgelegd. Tevens worden de bewerkingstijd en een instructietekst vastgelegd.

Een bewerking wordt uitgevoerd op een bepaald type workstation (workstationtypecode, naam, insteltijd). Een workcenter (workcentercode, naam) is opgebouwd uit een aantal workstations van een bepaald type. Er wordt bijgehouden welk type in welk workcenter staat. Een bepaald type kan in meer workcenters voorkomen.

Een bewerking heeft betrekking op een samenstelling van producten. Een samenstelling kan meer bewerkingen ondergaan. Een bewerking kan meer samenstellingen bewerken. Hierbij wordt een product geplaatst op een ander product. Van deze plaatsing wil men een instructietekst vastleggen, evenals het aantal van de samenstelling dat moet worden

gebruikt in de bewerking. Zo kan het voorkomen dat op een printplaat 6 chips van type A moeten worden geplaatst. Hier kunnen 1 tot 6 bewerkingen voor nodig zijn. Op de productspecificatie is aangegeven hoe en waar het product dient te worden geplaatst. Deze productspecificatie wordt als bijlage meegezonden met de fabricageorder.

Uitwerking



Figuur 79 Informatiestructuurdiagram

Entiteittypebeschrijvingen

Entity type: Klant

Definition: Een afdeling die de order plaatst voor een product.

Identifier: klantcode

Description: klantcode, naam, adres, telefoonnummer.

Entity type: Order

Definition: De order van een klant voor een bepaald product. In één order wordt altijd maar één product besteld.

Identifier: ordernummer

Description: ordernummer, ordertype, status, aantal, gevraagde leverdatum, gevraagde capaciteit.

Entity type:	Product
Definition:	Een product kan een component, een halffabrikaat of een eindproduct zijn. Hierin zijn opgenomen de logistieke gegevens van een product.
Identifier:	productnummer
Description:	productnummer, omschrijving, laatst geproduceerd op (datum), totaal geproduceerd aantal.
Entity type:	Samenstelling
Definition:	De relatie (samenstelling) tussen twee producten; wordt ook wel de stuklijstregel genoemd.
Identifier:	productnummer (parent), productnummer (child), prefix
Description:	prefix, aantal.
Entity type:	Routing
Definition:	De routingkop van de 'routing'.
Identifier:	routingnummer
Description:	routingnummer, effectieve startdatum, effectieve einddatum.
Entity type:	Routingregel
Definition:	De volgorde van bewerkingen binnen één workcenter.
Identifier:	routingnummer, workcentercode, regelvolgnummer
Description:	regelvolgnummer, instructietekst.
Entity type:	Workcenter
Definition:	Een verzameling van workstations.
Identifier:	workcentercode
Description:	workcentercode, naam.
Entity type:	Workstationtype
Definition:	De verzameling van gelijksoortige workstations.
Identifier:	workstationtypecode
Description:	workstationtypecode, naam, insteltijd.
Entity type:	Bewerking
Definition:	De bewerking binnen de routing per workcenter. De bewerking wordt uitgevoerd op een bepaald workstationtype.
Identifier:	routingnummer, workcentercode, regelvolgnummer, bewerkingsvolgnummer
Description:	bewerkingsvolgnummer, bewerkingstijd, bewerkingsinstructietekst.
Entity type:	Plaatsing
Definition:	De relatie tussen de stuklijstregel en de bewerking, waarin is opgenomen de instructietekst en het aantal te plaatsen componenten. De instructietekst geldt voor machines (bijvoorbeeld X- en Y-coördinaten) of voor mensen.
Identifier:	routingnummer, workcentercode, regelvolgnummer, bewerkingsvolgnummer, productnummer, productnummer, prefix

Description: plaatsingsaantal, instructietekst.

Identificerende attributen

(zie Identifier)

Verplichte attributen

Klant: klantcode, naam, telefoonnummer

Order: ordernummer, ordertype, status, aantal

Product: productnummer, omschrijving

Samenstelling: productnummer (parent), productnummer (child), prefix, aantal

Routing: routingnummer, effectieve startdatum, effectieve einddatum

Routingregel: routingnummer, workcentercode, regelvolgnummer

Workcenter: workcentercode, naam

Workstationtype: workstationtypecode, naam, insteltijd

Bewerking: routingnummer, workcentercode, regelvolgnummer, bewerkingsvolgnummer, bewerkingstijd

Plaatsing: routingnummer, workcentercode, regelvolgnummer, bewerkingsvolgnummer, productnummer, productnummer, prefix, plaatsingsaantal

En de verplichte relatietypen (minimumcardinaliteit = 1) uit het nformatiestructuurdiagram.

Validatieregels

Toegestane waarden voor ordertype: 'A', 'B', 'C'.

Toegestane waarden voor orderstatus: 'gepland', 'vrijgegeven', 'in-productie', 'onderbroken', 'gesloten'.

Afhankelijkheidsregels

Een order wordt afgehandeld volgens een routing die geldig is voor het product dat in die order wordt besteld.

Een bewerking kan alleen op die samenstellingen uitgevoerd worden ('bewerken') waarvoor geldt dat de samenstelling deel uitmaakt van het product en de routing – waar de bewerking een onderdeel van is – die voor dat product geldig is.

Een bewerking kan alleen worden uitgevoerd op die workstationtypen die geplaatst zijn in het workcenter opgenomen in de routing waarvan de bewerking een onderdeel vormt.

Afleidingsregels

Product is voor het laatst geproduceerd op = actuele afleverdatum van de laatste order (op het moment dat de orderstatus := 'gesloten').

Totaal geproduceerd aantal van product = totaal geproduceerd aantal van product + actuele afleveraantal van de order (op het moment dat de orderstatus := 'gesloten').

Het totaal van de gebruikte hoeveelheden van plaatsingen op een samenstelling moet gelijk zijn aan het aantal van die samenstelling.

De gevraagde capaciteit van een order wordt berekend aan de hand van de bewerkingstijd en de insteltijd van een routing.

Transitieregels

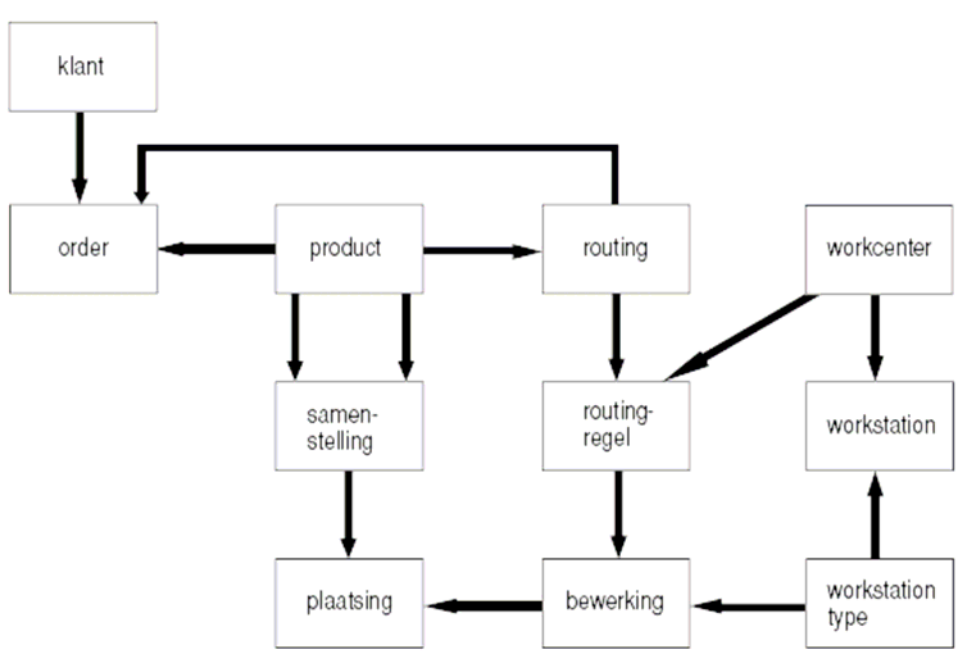
De initiële orderstatus is ‘gepland’ en de laatste orderstatus is ‘gesloten’. Toegestane transitie voor orderstatus zijn:

- gepland → vrijgegeven
- vrijgegeven → in-productie
- in-productie → onderbroken
- in-productie → gesloten.

B.3 Het specificatiemodel

Maak uitgaande van het informatiemodel een relationeel gegevensmodel voor de fabriek. Definieer de regels op het gegevensmodel.

Uitwerking



Figuur 80 Gegevensstructuur

Klant	Klant (<u>klantcode</u> , naam, adres, telefoonnummer)
Order	Order (<u>ordernummer</u> , ordertype, status, aantal, gevraagde leverdatum, gevraagde capaciteit, <klantcode>, <productnummer>, <routingnummer>)

Product	Product (<u>productnummer</u> , omschrijving, laatst geproduceerd op (datum), totaal geproduceerd aantal)
Samenstelling	Samenstelling (<parent_productnummer>, <child_productnummer>, prefix, aantal)
Routing	Routing (routingnummer, effectieve startdatum, effectieve einddatum, <productnummer>)
Routingregel	Routingregel (<routingnummer>, <workcentercode>, regelvolgnummer, instructietekst)
Workcenter	Workcenter (<u>workcentercode</u> , naam)
Workstationtype	Workstationtype (<u>workstationtypecode</u> , naam, insteltijd)
Workstation	Workstation (<workcentercode>, <workstationtype>)
Bewerking	Bewerking (<routingnummer, workcentercode, regelvolgnummer>, <u>bewerkingsvolgnummer</u> , bewerkingstijd, bewerkingsinstructietekst, <workstationtypecode>)
Plaatsing	Plaatsing (<routingnummer, workcentercode, regelvolgnummer, <u>bewerkingsvolgnummer </u>

Constraints

Ordertype = {A, B, C}

Orderstatus = {gepland, vrijgegeven, in-productie, onderbroken, gesloten}

Enkele event-getriggerde regels

on: PRE-INSERT, PRE-UPDATE van Order

if: Order.status not NULL

then: Check: toegestane transitities:

gepland → vrijgegeven

vrijgegeven → in-productie

in-productie → onderbroken

in-productie → gesloten.

on: POST-CHANGE van Order.status

if: Order.status = gesloten

then: Do: Product.laatst_geproduceerd_op := Order.actuele_afleverdatum

on: POST-CHANGE van Order.status

```

if: Order.status = gesloten
then: Do: Product.totaal_geproduceerd_aantal:= Product.totaal_geproduceerd_aantal +
      Order.actuele_afleveraantal

```

B.4 Een regel nader uitgewerkt

In de fabriek wordt per dag een pakket van orders uitgevoerd. De gevraagde capaciteit van het orderpakket wordt vergeleken met de beschikbare capaciteit. In geval van over- of onderbelading worden orders uitgesteld of vervroegd.

Geef mogelijke oplossingsrichtingen aan voor de capaciteitsberekening. Maak hierbij gebruik van de onderstaande beschrijving.

Capaciteitsberekening

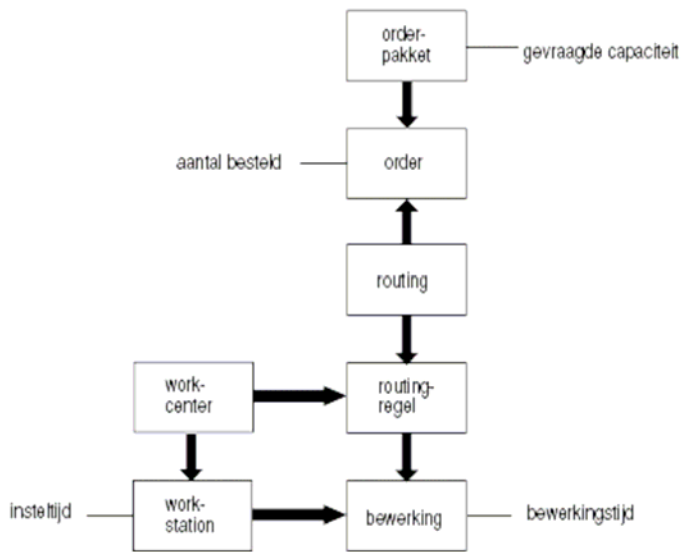
De gevraagde capaciteit voor een orderpakket wordt berekend uit de gevraagde capaciteiten van de afzonderlijke orders. De gevraagde capaciteit van een order wordt berekend aan de hand van de insteltijden en bewerkingstijden van de routing. Deze tijden worden per eenheid product berekend.

De capaciteitsberekening bestaat uit drie stappen:

1. Bereken tijden per routing:
 Bereken de bewerkingstijd en de insteltijd per routing volgens:
 bewerkingstijd = S bewerkingstijden van de in de routing opgenomen bewerkingen
 insteltijd = S insteltijden van de in de routing opgenomen workstations.
2. Bereken capaciteit per order:
 Bereken de gevraagde capaciteit van de order volgens:
 gevraagde capaciteit = orderhoeveelheid * bewerkingstijd van de gekozen routing +
 insteltijd van de gekozen routing.
3. Bereken capaciteit per dag:
 Bereken de gevraagde orderpakketcapaciteit per dag volgens:
 gevraagde capaciteit = Totaal gevraagde capaciteit van de orders voor die dag.

Uitwerking

De eerste oplossingsstrategie bestaat uit één programma (één regel) die de gevraagde capaciteit van een orderpakket berekent. Het resultaat is gekoppeld aan een virtueel veld of een database-veld. Het programma wordt gestart wanneer het antwoord op de vraag 'Wilt u rekenen?' 'Ja' is. Figuur 81 toont de voor deze oplossing relevante attributen.



Figuur 81 Het gegevensmodel voor de één-proces-oplossing

Het programma om de gevraagde capaciteit van een orderpakket te berekenen ziet er in SQL als volgt uit:

```

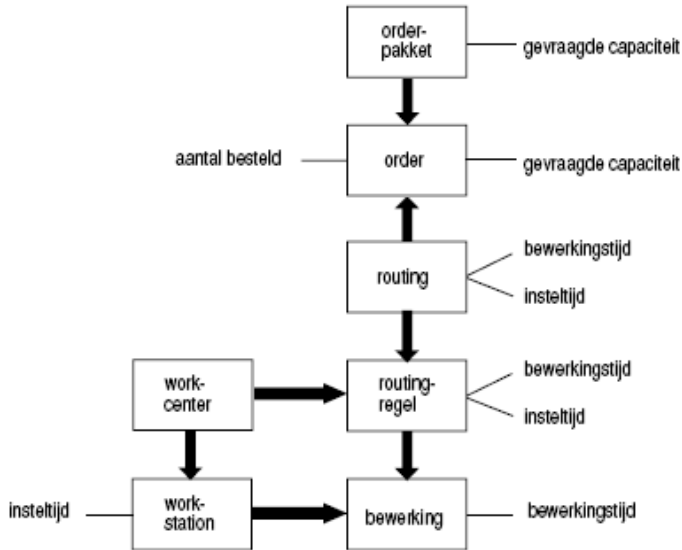
SELECT  SUM (Bewerking.bewerkingstijd * Order.aantal),
        SUM (Workstation.insteltijd)
FROM    Workstation,
        Bewerking,
        Routingregel,
        Order,
        Orderpakket
WHERE   Bewerking.workstation_id = Workstation.workstation_id
AND     Bewerking.routingregel_id = Routingregel.routingregel_id
AND     Routingregel.routing_id = Order.routing_id
AND     Order.datum = Orderpakket.datum
AND     Orderpakket.datum = jjmdd.

```

De tweede oplossingsstrategie bestaat uit het distribueren van de regels. Deze oplossing bestaat uit twee stappen:

- Verdeel de capaciteitsberekening in de vorm van informatieregels over het gegevensmodel.
- Bepaal vervolgens de triggering van de informatieregels.

Figuur 82 toont de voor deze oplossing relevante attributen.



Figuur 82 Gegevensmodel voor de gedistribueerde oplossing

Hierop worden de volgende regels gedefinieerd:

1. gevraagde capaciteit van een orderpakket = S gevraagde capaciteiten van de orders van die dag;
2. gevraagde capaciteit van een order = insteltijd van de gekozen routing + bewerkingstijd van de gekozen routing * orderaantal;
3. bewerkingstijd van een routing = S bewerkingstijden van de bijbehorende routingregels;
4. insteltijd van een routing = Σ insteltijden van de bijbehorende routingregels;
5. bewerkingstijd van een routingregel = S bewerkingstijden van de bijbehorende bewerkingen;
6. insteltijd van een routingregel = S insteltijden van de workstations waarop de bewerkingen worden uitgevoerd.

Het totale rekenproces wordt in drie delen opgestart:

1. De routingtijden per eenheid product worden berekend na het wijzigen van de insteltijd van een workstation of de bewerkingstijd van een bewerking, of na het toevoegen of verwijderen van een bewerking.
2. De ordercapaciteit wordt berekend na het toewijzen van een routing aan de order.
3. De gevraagde capaciteit van een orderpakket wordt berekend na expliciet aangeven van de gebruiker: antwoord 'Ja' op de vraag 'Wilt u rekenen?'.

De complete specificatie van de event-getriggerde regels is als volgt:

- (1) on: RE-INSERT, PRE-UPDATE van `Bewerking`
PRE-INSERT, PRE-UPDATE van `Bewerking.bewerkingstijd`
PRE-INSERT, PRE-UPDATE van `Workstation.insteltijd`
if:
then:

```
SELECT  SUM insteltijd,
        SUM bewerkingstijd
FROM    Workstation,
        Bewerking,
        Routingregel
WHERE   Bewerking.workstation_id = Workstation.workstation_id
AND     Bewerking.routingregel_id = Routingregel.routingregel_id
AND     Routingregel.routing_id = &routing_id.

SELECT  SUM bewerkingstijd,
        SUM insteltijd
FROM    Routingregel,
        Routing
WHERE   Routingregel.routing_id = Routing.routing_id
```
- (2) on: PRE-INSERT, PRE-UPDATE van `Order`
if: `Order.routing_id` not NULL
then:

```
SELECT  &aantal * routing.bewerkingstijd + routing.insteltijd
FROM    Routing
WHERE   Routing.routing_id = Order.routing_id
AND     Routing.routing_id = &routing_id.
```
- (3) on: POST-CHANGE van het antwoord (ja/nee)
if: `antwoord = ja`
then:

```
SELECT  SUM (order.gevraagde_capaciteit)
FROM    Order
WHERE   Order.datum = jjmmdd.
```

Literatuur

- [Bakema] Volledig Communicatiegeoriënteerde Informatiemodellering (FCO-IM), G. Bakema, J.P. Zwart en H. van der Lek, Kluwer Bedrijfswetenschappen, 1996.
- [Booch] The Unified Modeling Language User Guide, G. Booch, J. Rumbaugh en I. Jacobson, Addison-Wesley, 1999.
- [Bosschers] Handboek Projectmanagement, de Tipi-approach, E. Bosschers e.a., Thema, 2002.
- [Date] An introduction to Database Systems, C.J. Date, Addison-Wesley Publishing Company, Fifth edition, 1990.
- [DSDM] Dynamic Systems Development Methodology, DSDM Consortium, Tesseract Publishing, 1997.
- [Griethuysen] INFOMOD, een samenvatting, J.J. van Griethuysen, D.A. Jardine, Academic Service, 1988.
- [ISO] Concepts and Terminology for the Conceptual Scheme and the Information Base, International Organization for Standardization, Working Group TC97/SC5/WG3, Report ISO/TC97/TR9007, 1987.
- [Jacobson] The Unified Software Development Process, I. Jacobson, G. Booch en J. Rumbaugh, Addison-Wesley, 1999.
- [Kranenburg] De moderne softwarefabriek – de organisatie en werking van een Software Development & Maintenance Center, K. Kranenburg, F. Nelissen en J. Brouwer, ten Hagen & Stam Uitgevers, 2e herziene druk, 2003.
- [Marca] SADT – Structured Analysis and Design Technique, D.A. Marca, C.L. McGowan, McGraw-Hill, 1987.
- [Martin] Rapid Application Development, J. Martin, McMillan Publishing Company, 1991.
- [Peeters] Objectgeoriënteerde domeinanalyse – Een systematische aanpak voor het ontwerp van UML-klassendiagrammen volgens EXPO, F. Peeters en F. Vonken, Academic Service, 2001.
- [Pollaert] Informatie analyse – De brug van bedrijfsdoelen naar ICT oplossingen, W. Pollaert en K. Ruigrok, Thema, 2002.

- [Pollaert] Database ontwerp, W. Pollaert F. van Eeden en K. Kranenburg, Thema, 2004.
- [de Rooij] NIAM, T. de Rooij, Kluwer Bedrijfswetenschappen, 1995.
- [Wintraecken] Informatie-analyse volgens NIAM in theorie en praktijk, J.J.V.R. Wintraecken, Academic Service, 1987.