Object-georiënteerde systeemontwikkeling Een trend in de toekomst?

A.C.M. Bartels-Mertens, A.J.M. Beulens,
C.A. van Loon, D. Schuyl-Laros

Haagse Hogeschool, sector Informatica
Louis Couperusplein 19
2514 HP 's-Gravenhage
tel. 070-3618506
fax. 070-3618599
email kees@si.hhs.nl

Referaat

Op basis van een analyse van problemen bij de huidige systeemontwikkeling wordt een aantal criteria afgeleid waaraan (ook) de object-georiënteerde benadering kan worden getoetst. De resultaten van deze toets wordt beschreven voor de in dit artikel geschetste object-georiënteerde benadering. Als afsluiting wordt gekeken naar gevolgen van invoering van de object-georiënteerde benadering voor de huidige systeemontwikkeling.

Trefwoorden:
object-oriëntatie, object-georiënteerde systeemontwikkeling

Inleiding

In organisaties volgen ontwikkelingen en trends op het gebied van management van de informatievoorziening, de ontwikkeling van geautomatiseerde informatiesystemen en het gebruik van informatietechnologie elkaar snel op. Sommige blijken van tijdelijke, andere weer van structurele aard. Een van die trends is de object-georiënteerde benadering bij het ontwikkelen van informatiesystemen, database management systemen en databases. Deze informatietechnologische trend kan mogelijkheden bieden om de informatievoorziening beter af te stemmen op de behoefte van de organisatie. Door de toenemende eis tot systeemintegratie op zowel organisatorisch als bedrijfstakniveau en de verbreiding naar en over functionele gebieden, wordt het ontwikkelen van informatiesystemen steeds complexer van aard, waardoor aanwijsbare knelpunten zijn ontstaan bij de huidige benadering van het effectief en efficiënt ontwikkelen van informatiesystemen. Een deel van deze knelpunten kan als decompositievraagstukken worden gezien, waarvoor de oplossing gezocht moet worden in een betere keuze van de optimale abstractie- en detailleringsniveaus. De vraag is of de object-georiënteerde benadering bij het ontwikkelen van informatiesystemen de huidige knelpunten kan oplossen.

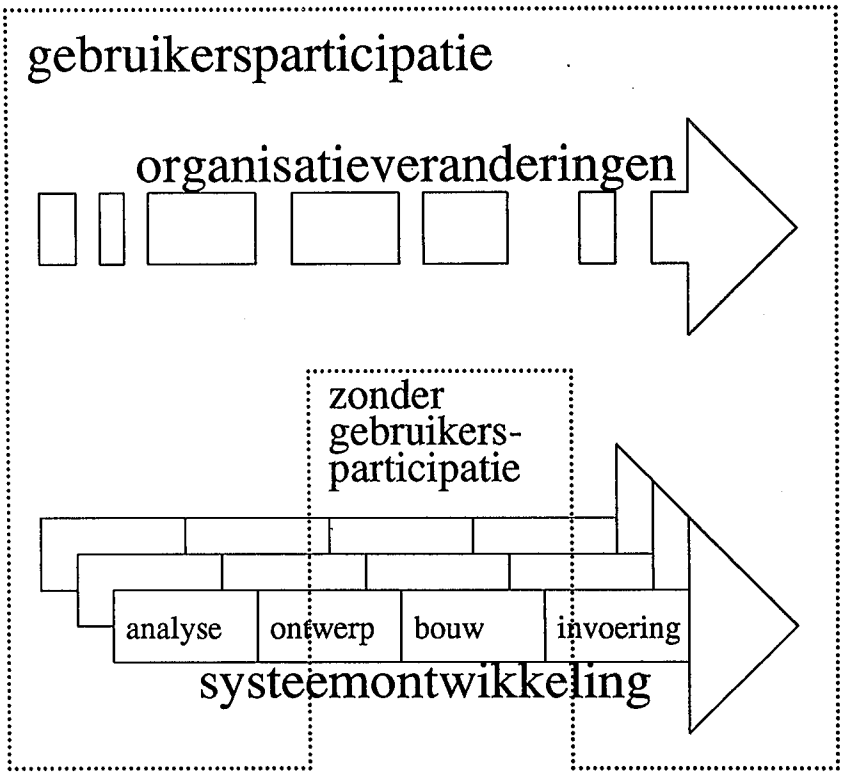
In dit artikel wordt in paragraaf 1 een overzicht gegeven van de problemen bij de huidige systeemontwikkeling. Paragraaf 2 geeft een beschrijving van de object-georiënteerde benadering. In paragraaf 3 worden vanuit de essentie van een aantal ontwerpfilosofieën de

van een aantal ontwerpfilosofieën de criteria afgeleid, waaraan methoden moeten voldoen, waarna in paragraaf 4 de toetsing van de object-georiënteerde benadering hieraan volgt. Tenslotte worden in paragraaf 5 de gevolgen, die de object-georiënteerde benadering voor de huidige systeemontwikkeling met zich mee zou kunnen brengen, beschreven.

1 Huidige systeemontwikkeling

De huidige systeemontwikkelmethoden kenmerken zich veelal door een stapsgewijze verfijning (globaal-detail) met als uitgangspunt de samenhang tussen de processen en de gegevens, waarbij gegevensverwerkende processen gegevens bewerken en creëren. Een methode dient tijdens het ontwikkelproces effectieve ondersteuning te bieden en voldoende aanknopingspunten te geven voor een efficiënt beheer van het ontwikkelprojekt. Om aan deze eisen te voldoen, wordt een gefaseerde aanpak toegepast waarin alle facetten van onder andere communicatie, decompositie, projektmanagement en documentatie aan de orde komen (Bemelmans, 1987). Binnen de huidige ontwikkelcyclus onderkent (Bemelmans, 1987) twee stromingen. De ene school baseert de systeemontwikkeling op het zogenaamde waterval model, bestaande uit een lineaire serie van activiteiten (analyse, ontwerp, bouw en onderhoud), waarbij de voorgaande activiteit eerst voltooid moet zijn voordat de volgende begint. Hierbij worden de systeemspecificaties gedurende het ontwikkelproces in een vroeg stadium vastgelegd, waardoor deze methode bij organisaties met een sterk veranderende informatiebehoefte minder past. Een aanpak die volgens (Bemelmans, 1987) beter past bij evolutionaire systeemontwikkeling is prototyping. In een relatief korte doorlooptijd worden de informatiebehoeften, in samenwerking met de gebruikers, vertaald in een prototype van een deel van het informatiesysteem, waarna het systeem geleidelijk aan verbeterd en uitgebreid wordt.

Om de efficiency, effectiviteit en de beheersbaarheid van de informatievoorziening te verbeteren, de kwaliteit te verhogen en de systeemintegratie te bevorderen, zijn veel organisaties overgegaan tot het formuleren van een informatiebeleid en het concretiseren van dit beleid in een

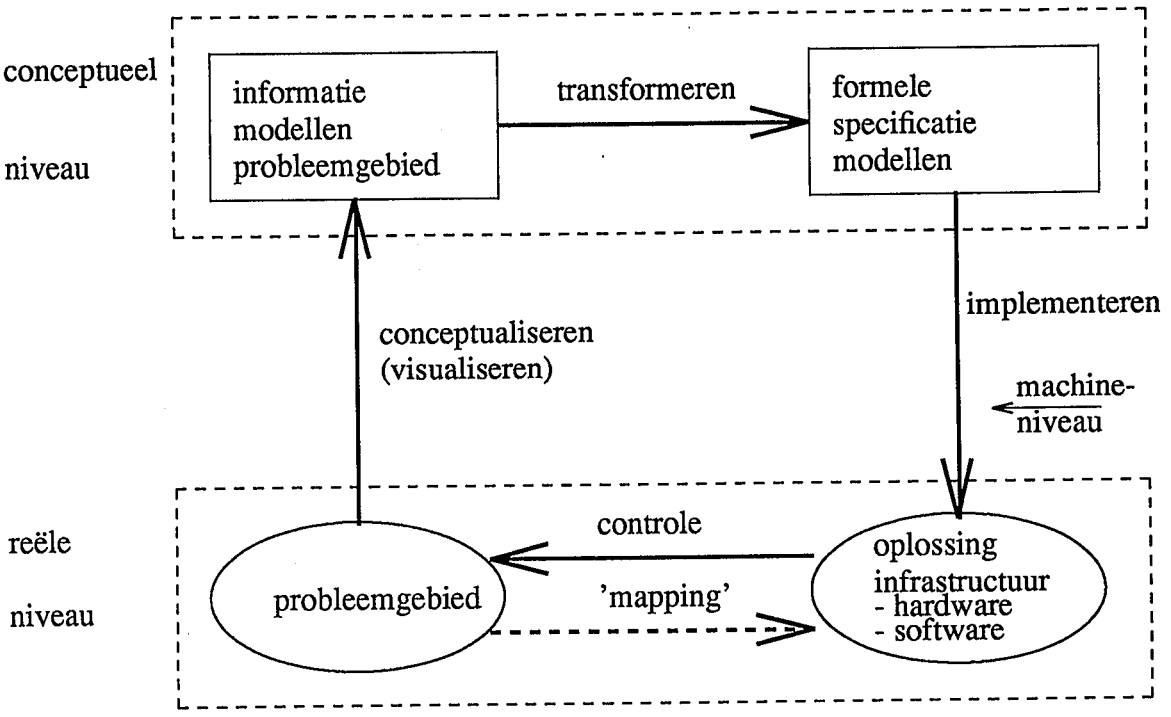


Figuur 1.1
samenhang tussen organisatie veranderingen en het ontwikkelen van informatiesystemen

informatieplan, waarbinnen het ontwikkelproces wordt gedaan. Tijdens het ontwikkelproces spelen verschillende aspecten van gegevens een rol wat aanleiding geeft tot het ontwerpen van verschillende modellen. In deze modellen komt het waarom, het wat, het hoe van informatie en de samenhang tussen de gegevensaspecten tot uiting (Bemelmans, 1987). De afbeelding van de modellen van het besturingssysteem (waarom) zijn vaak niet in overeenstemming met de besturingssituatie van de organisatie, waardoor het uitgangspunt niet goed tot zijn recht komt. Om de complexiteit van het systeem te reduceren, wordt een top-down functioneel gerichte werkwijze gehanteerd. Hierbij ontstaat een toenemende detaillering waarbij andere aspecten worden bekeken en indien nodig toegevoegd. Bij het hanteren van de hierboven beschreven denk- en werkwijze zijn tijdens het uitvoeren van de verschillende fasen de volgende problemen gesignaleerd en in de literatuur beschreven.

1.1 Problemen in analyse- en ontwerpfase

In (Gupta, 1988) wordt gesteld dat de meeste fouten bij het vaststellen van de gebruikersbehoeften ontstaan door gebrekkige communicatie tussen analisten en gebruikers. Daar deze fouten pas in een laat stadium aan het licht komen, blijken zij tevens het kostbaarste te zijn om te herstellen. In (Bemelmans, 1986) wordt aangegeven dat het vaststellen van criteria voor opdeling in subsystemen en de keuze van het juiste abstractie- en detailleringsniveau in de praktijk moeilijk is. Bij een te globale beschrijving zullen zaken over het hoofd worden gezien, waardoor een goede basis voor de realisatie van een informatiesysteem ontbreekt. Bij een te gedetailleerde specificatie kan veelal door de beperkte ontwikkelcapaciteit, slechts een beperkt aantal deelsystemen worden uitgewerkt wat consequenties voor de organisatie heeft. Bij een beperkte uitwerking van het aantal deelsystemen is de garantie dat hierdoor kwalitatief goede informatiesystemen worden geleverd, niet gewaarborgd. Verder wordt in (Bemelmans, 1986) aangegeven dat het formuleren van functionele en prestatie eisen voor de informatievoorziening en voor specifieke informatiesystemen onder andere afhangt van de organisatie, de besturingscontext en de omgeving. Hierdoor is het formuleren van dynamisch robuuste functionele en prestatie eisen voor een informatiesysteem erg moeilijk. De



Figuur 2.1.
Conceptualisatie, transformatie en implementatie

analysefase tracht de gebruikerseisen vast te leggen, waarna deze in de ontwerpfase getransformeerd worden naar formele specificaties. Hierbij is gebleken dat de kloof tussen analyse- en ontwerpfase zo groot is, dat fouten in het uiteindelijke resultaat kunnen ontstaan. Een dynamische omgeving veroorzaakt voortdurend organisatorische veranderingen. Als gevolg hiervan kan het gebeuren dat de eisen gesteld aan een informatiesysteem veranderen tijdens het ontwikkeltraject, waardoor een informatiesysteem ten tijde van de invoering niet meer voldoet aan de meest actuele eisen (zie figuur 1.1). Door de toenemende complexiteit van de omgeving en de eis tot systeemintegratie (technische infrastructuur en gegevensarchitectuur) op zowel organisatorisch als bedrijfstakniveau, is de materie die men tracht te automatiseren moeilijk met de huidige benaderingswijze te analyseren.

1.2 Problemen in bouw- en onderhoudsfase

Bij het bestuderen van de literatuur over ontwikkelingen binnen de software, zoals bijvoorbeeld de object-georiënteerde benadering, wordt vaak de term 'software crisis' als aanleiding voor veranderingen gegeven. De opvattingen over de oorzaak van deze crisis lopen uiteen van de steeds geringer wordende persoonlijke effectiviteit van de systeemontwikkelaar tot de groeiende hoeveelheid software die onderhouden moet worden en de toenemende complexiteit van de hedendaagse systemen. Nog steeds wordt aangegeven dat een duidelijke taakverschuiving waarneembaar is van de automatiseringsafdeling van nieuwbouw naar onderhoud. (Gupta, 1988) geeft aan dat 80% van de tijd van het automatiseringspersoneel aan onderhoud wordt besteed en dat hiervoor 50% beslag wordt gelegd op het totale automatiseringsbudget. Goede courante cijfers ten aanzien van de totale arbeidstijd en kosten die hiermee gepaard gaan ontbreken.

2 Object-georiënteerde benaderingswijze

In deze paragraaf wordt een beschrijving gegeven van object-georiënteerde systeemontwikkeling.

2.1 Object-georiënteerde systeemontwikkeling

Het terrein van de object-georiënteerde systeemontwikkeling beperkt zich gewoonlijk tot een informatiesysteem en het deel van de organisatie (het reële probleemgebied) dat door het informatiesysteem wordt beïnvloed. Doordat het eindresultaat van het systeemontwikkelproces een formeel systeem (infrastructuur: hard- en software) is, dienen de problemen gesignaleerd in het probleemgebied geformaliseerd en opgelost te worden. Met andere woorden, om de kloof te dichten tussen de organisatie met zijn informatieproblemen enerzijds en de oplossing van deze problemen, een operationeel informatiesysteem anderzijds, wordt er gebruik gemaakt van verschillende conceptuele object-georiënteerde modellen (zie figuur 2.1). De rol van deze conceptuele object-georiënteerde modellen tijdens dit proces is tweeledig en geven een viertal aspecten (waarom, wat, hoe en waarmee) van informatie en de samenhang hiertussen weer, die in onder andere (Bemelmans, 1987), (Welke, 1977) en (Sundgren, 1978) worden beschreven. In de eerste plaats de rol van de informatiemodellen. Deze kan omschreven worden als het aangeven van de besturingssituatie (waarom) en de manier waarop deze gepland, gecontroleerd en ondersteund wordt. Tevens dienen de gegevens en de

gegevensverwerkende processen zodanig weergegeven te worden dat de gebruikersinformatie eisen (wat) duidelijk naar voren komen. In de tweede plaats de rol van de formele specificatie modellen. Hierbij wordt het probleem opgelost (hoe en waarmee). Deze modellen geven zowel de logische als de fysieke view van het te bouwen informatiesysteem aan, met als eindresultaat een operationeel informatiesysteem.

Het besturingsparadigma kan van dienst zijn om binnen deze kaders te komen tot model- en informatiesysteemontwikkeling, waarbij de nadruk ligt op de doeluitvoer, de daarmee samenhangende invoer en de informatieuitwisseling. In het besturingsparadigma wordt als model van een objectsysteem een bestuurd systeem, een besturend systeem en een omgeving onderkend. Om aan de voorwaarden van effectieve besturing te voldoen, dient het besturend systeem te beschikken over het doel, een model, informatie over de toestand van en gebeurtenissen in het bestuurd systeem en over voldoende besturingsvariëteit. Om de omvangrijke en complexe besturing te kunnen beheersen, kan een van de volgende twee strategieën ontleend aan (Galbraith, 1976) toegepast worden:

a verminder de behoefte aan coördinatie door speling in de organisatie in te bouwen en door autonome taken te scheppen; of
b verhoog het coördinatie vermogen door te investeren in verticale informatiesystemen en laterale relaties.

Om de beheersbaarheid van de informatievoorziening te verhogen en het juiste abstractie- en detailleringsniveau aan te geven, zal van de organisatie en van ieder aspect binnen de organisatie de besturingssituatie in objectsysteemmodellen tot uiting dienen te komen, waarbij de te onderkennen aspecten vanuit de organisatiemodellering duidelijk dienen te worden aangegeven. Hierbij kan als decompositiecriterium de in de systeemleer onderscheiden aspectsystemen genomen worden.

Ieder objectsysteemmodel (bedrijfsmodel) kan vanuit een drietal invalshoeken in een drietal modellen worden aangegeven (Waes, 1991) afhankelijk van wat men wenst te beschrijven. Ten eerste de proces-georiënteerde invalshoek (het procesmodel). Hierin wordt de functionele structuur (processen) aangegeven en de manier waarop deze worden gepland, gecontroleerd en ondersteund (Waes, 1991) (zie figuur 2.2). De data-georiënteerde invalshoek (het objectmodel) geeft de objectklassen weer die relevant zijn voor het beschouwingsgebied. De communicatie-georiënteerde invalshoek (de relatie) geeft het verband aan tussen het proces- en objectmodel weer.

De organisatiemodellering (zie figuur 2.2) levert modellen op met hun onderlinge relaties waarin de processen, de structuren, de procedures en de cultuur van de organisatie tot uiting komen. Een mogelijke benaderingswijze is die van Mintzberg (Mintzberg, 1979), waarin een organisatie wordt beschreven als een combinatie van formele structuur (organogram), primaire processen (en de manier waarop deze worden gecontroleerd en gemanaged), informele structuur, samenwerkings- en besluitvormingsstructuur.

De dynamische omgeving waarin het probleemgebied zich bevindt, veroorzaakt voortdurend organisatorische veranderingen (zie figuur 1.1), die van invloed kunnen zijn op de organisatiemodellen. Gelet op de nauwe samenhang tussen organisatiemodellering en objectsysteemmodellering dienen wijzigingen in organisatiemodellen te worden doorgegeven aan het desbetreffende objectsysteemmodel (figuur 2.2), waardoor het conceptuele informatiemodel van de besturingssituatie (waarom) kan worden aangepast aan de actuele situatie van de organisatie.

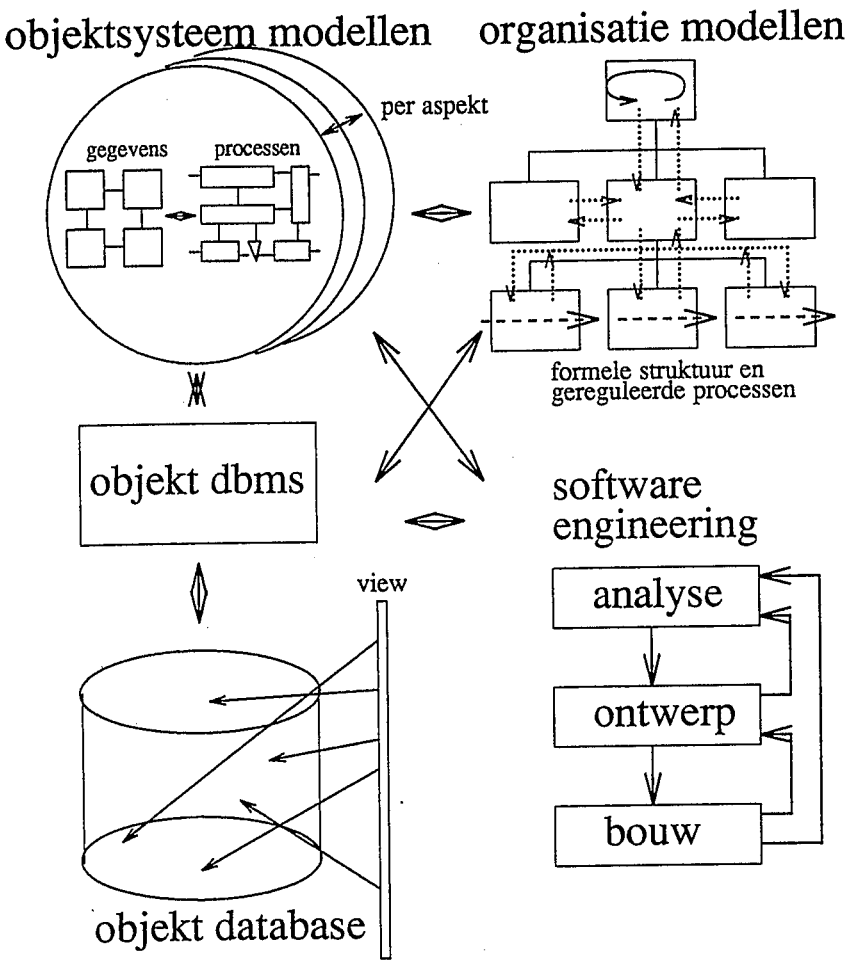
De objectsysteem- en organisatiemodellen worden door een structureringsmechanisme per aspectsysteem in de bedrijfsomvattende object database opgeslagen, waardoor de basis wordt gelegd voor een objectklassen-bibliotheek. Om de objecten op een gestructureerde wijze op te kunnen slaan, worden de eisen gesteld aan een object database door (Abbenhardt, 1990) als volgt omschreven:

- er dienen opslagmogelijkheden te zijn voor complexe objecten (eigenschappen en acties) met de techniek voor abstractie (aggregatie) en classificatie (generalisatie/specialisatie);
- er mogen geen beperkingen voor granulariteit en dynamische modificatie met consistentie bewaking voor deze objectklassen bestaan. Dit stelt zeer hoge en zware eisen aan een Data Base Management Systeem (DBMS).

De Software Engineering (SE) kan met behulp object-georiënteerde technieken, een of meerdere informatiesystemen ontwikkelen. Het uitgangspunt voor SE wordt gevormd door die objectsysteemmodellen (zowel per (deel-)organisatie als per aspect) die door het te ontwikkelen informatiesysteem worden beïnvloed om zodoende het juiste abstractie- en detailleringsniveau aan te geven. Tijdens het ontwikkelproces dient de ontwikkelaar gebruik te maken van eerder gedefinieerde objectklassen, die relevant zijn voor het probleemgebied, om zodoende bestaande definities van objectklassen uit de bedrijfsomvattende object database aan te passen dan wel nieuwe toe te voegen. Hierbij zal een kopie van de geselecteerde standaardobjectklassen worden gemaakt naar de projectdatabase. Hierna worden de objectklassen aangepast en in nieuwe toestand weer opgeslagen in de bedrijfsomvattende object database. Door deze wijze van werken zou de produktiviteit van de systeemontwikkeling verhoogd kunnen worden. Alle objectklassen, die in de verschillende projecten worden aangepast dan wel toegevoegd, worden in de bedrijfsomvattende object database opgeslagen, waarbij de consistentiebewaking door het Data Base Management Systeem (DBMS) dient te worden verzorgd. Zie figuur 2.2 voor de samenhang tussen organisatiemodellering, objectsysteemmodellering, Database Management Systeem (DBMS) met de bedrijfsomvattende object database en de Software Engineering (SE).

De discussies rond de begrippen object en objectklasse zijn talloos. De volgende definitie zal in dit artikel worden gehanteerd en is afgeleid van de omschrijving die (Shlaer, 1988) geeft:

Een objectklasse is een abstractie van een set van dingen (elementen) uit de werkelijkheid waarvoor geldt dat:



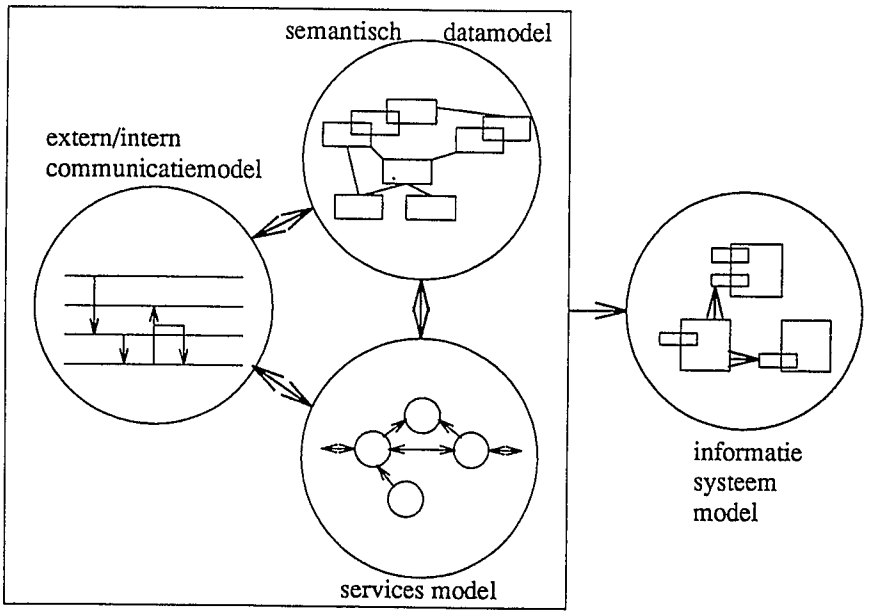
Figuur 2.2.
De samenhang tussen de organisatiemodellen, objectsysteemmodellen, object-database, DBMS en SE

- al die elementen binnen een objectklasse dezelfde eigenschappen, kenmerken en gedrag dienen te hebben en
- al die elementen binnen een objectklasse onderworpen zijn en beantwoorden aan dezelfde regels.

Een object wordt gedefinieerd als een instance van een objectklasse. In onder andere (Katwijk, 1990) en (Coad, 1990) wordt aangegeven dat de object-georiënteerde systeemontwikkeling zijn oorsprong vindt in object-georiënteerd programmeren aan de ene kant en in verschillende vormen van data-modellering aan de andere kant. Hierbij is de systeemontwikkeling erop gericht voor het objectsysteem, waarvoor een informatiesysteem moet worden ontwikkeld, objecten te identificeren en voor elk object in de reële wereld een of meer objectklassen in de programmatuur te construeren (Katwijk, 1990). De denk- en werkwijze van object-georiënteerde systeemontwikkelmethode wordt gekenmerkt door:

- het denken in bouwstenen en interactie tussen deze bouwstenen;
- het denken in objecten, kenmerken, eigenschappen en gedrag van die objecten;
- het abstraheren van objecten door middel van classificatie en aggregatie, waarmee de complexiteit van het probleemgebied in kaart wordt gebracht;
- communicatie via messages (berichten), hetgeen zowel voor de gebruiker, voor het systeem als voor de instances binnen het systeem geldt;
- het werken met en het beheersen van objectstrukturen en structuurovergangen door middel van classificatie van de implementatie aspecten, waardoor een kader ontstaat waarbij de inheritance (overerving) zichtbaar wordt (Page-Jones, 1990).

Om de effectiviteit te verhogen en voldoende aanknopingspunten te geven voor een efficiënt beheer van het ontwikkelproces wordt een gefaseerde aanpak toegepast. In contrast met de huidige systeemontwikkelmethoden, die het probleem trachten op te lossen door een stapsgewijze verfijning (globaal-detail), is de object-georiënteerde benadering gericht op het opdelen van het systeem in modules (bouwstenen), waarbij de inwendige structuur van de modules voor de decompositie niet van belang is, om zodoende de meest robuuste informatiesystemen te ontwikkelen. Hierbij kent de aanpak een iteratief karakter, waarbij de activiteiten in de verschillende fasen elkaar overlappen en in elkaar overgaan (Henderson-Sellers,



Figuur 2.3.
Een overzicht van de gebruikte modellen en hun relatie uit de OOA

1990), waardoor de fasen anders benaderd en benadrukt worden.

De object-georiënteerde systeemontwikkeling leidt tot een set van conceptuele modellen (informatiemodellen en formele specificatie modellen). Hierbij zal de object-georiënteerde analysefase (OOA) een set van informatiemodellen opleveren, waarin objecten, classificatie structuren, communicatie en services (functionaliteiten) tot uiting komen. De object-georiënteerde ontwerpfase (OOD) zal een set van formele specificatie modellen opleveren, waarbij een verdere invulling wordt gegeven aan de ontworpen modules (een of meerdere objectklassen), waardoor de inwendige structuur gestalte gaat krijgen. Deze modellen dienen een notatiewijze te hanteren waarin naast objectstrukturen en structuurovergangen ook messagesprotocolli, transacties, status en kennis (Rademackers, 1989) tot uiting komen, om zodoende de onderzochte materie te structureren. De complexiteit van het systeem wordt in kaart gebracht door het probleem te decomponeren met behulp van objecten en door middel van aggregatie en classificatie structuur aan te brengen, waardoor een hoger abstractie niveau wordt verkregen.

2.1.1 Object-georiënteerde analyse (OOA)

Het doel van de analysefase is de gebruikerseisen over wat het te ontwikkelen systeem moet doen te verzamelen en vast te leggen met behulp van conceptuele modellen, waarin de essentiële objectklassen, de relaties en de interacties tussen deze tot uiting komen. Het uitgangspunt van de OOA zijn die objectsysteemmodellen van de organisatie en per aspect (waarom), waarin de bedrijfskundige context van dat deel van de organisatie tot uiting komt, dat door het te ontwikkelen informatiesysteem wordt beïnvloed. Het resultaat van de OOA is een conceptueel object-georiënteerd informatiesysteemmodel vanuit zowel de statische als de dynamische semantiek.

2.1.2 Object-georiënteerd ontwerpen (OOD)

Het doel van het object-georiënteerd ontwerpen is het oplossen van het ontwerpprobleem met behulp van formele specificatie modellen. Het uitgangspunt van de OOD is het object-georiënteerde informatiesysteemmodel uit de OOA, waarin de objectklassen, het gedrag en de services (functionaliteiten) tot uiting komen. In de OOD zullen de module- en processtructuur worden aangegeven. Hierbij zal per module een verdere invulling worden gegeven aan de objectklassen, zodat de inwendige structuur per objectklasse meer gestalte krijgt. Hierbij dient zoveel mogelijk van aanwezige objectklassen gebruik gemaakt te worden. Indien bestaande objectklassen worden gebruikt, zal een beslissing genomen moeten worden of men het ontwerp aanpast, waarbij een terugkoppeling naar de analysefase noodzakelijk is, of dat de services (functionaliteiten) en objectklassen worden aangepast. Om tot een set van object-georiënteerde formele specificatiemodellen te komen kan een aantal stappen worden uitgevoerd. Deze zijn beschreven in o.a. (Bartels-Mertens, 1991), (Booch, 1990), (Coad, 1990), (Henderson-Sellers, 1990), (Jacobson, 1987), (Page-Jones, 1990) en (Smith, 1988).

2.1.3 Object-georiënteerd programmeren (OOP)

In object-georiënteerde programmeertalen wordt de structuur van de gehele applicatie gevormd door de objectklassen waarbinnen de data en de procedures worden ingekapseld (encapsulation). Het is mogelijk om via inheritance (overerving) objectklassen eigenschappen met elkaar te laten delen, waardoor via specialisatie (inheritance downward) of generalisatie (inheritance upward) nieuwe objectklassen gecreëerd kunnen worden. De messages (berichten) dienen als communicatiemiddel tussen de objectklassen onderling. Hierbij is polymorfisme toegestaan, wat inhoud dat aan messages (berichten) in verschillende objectklassen dezelfde naam gegeven kan worden. De methods (acties) zorgen voor de uitvoering van de bewerking. Dit in tegenstelling tot de niet object-georiënteerde programmeertalen, die uitgaan van processen (vastgelegd in programma's), die bijbehorende gegevens creëren en bewerken. De bouwfase zal vooral gekenmerkt worden door het creëren van nieuwe objectklassen uit bestaande objectklassen door deze te specialiseren dan wel combinaties hiervan te maken, waardoor nieuwe applicaties ontstaan. De nadruk bij het onderhoud zal liggen op het oplossen van fouten in methods (acties) en het toevoegen van functionaliteit door uitbreiden van messages (berichten) en daarbij behorende methods (acties).

3 Criteria

3.1 Inleiding

In (Bemelmans, 1987) wordt een ontwikkelmethode omschreven als een denk- en werkwijze voor de ontwikkeling van informatiesystemen. De denkwijze (ontwerpfilosofie) achter de methode weerspiegelt de manier waarop men tegen de werkelijkheid aankijkt en heeft direct gevolgen voor de wijze waarop men te werk zal gaan. De meest voorkomende ontwerpfilosofieën worden in (Bemelmans, 1987) als volgt beschreven:

- procesgeoriënteerd versus gegevensgeoriënteerd: Bij de procesgeoriënteerde aanpak worden de bedrijfsactiviteiten en gegevensverwerkende processen als uitgangspunt gekozen. De gegevensgeoriënteerde benadering begint met het analyseren en structureren van de noodzakelijke gegevens;
- top down versus bottom up: Top-down betekent dat men eerst een ontwerp in hoofdlijnen maakt en daarna detailleert. Bottom up daarentegen begint met het detail en werkt zodoende naar de hoofdlijnen toe;
- totaalaanpak versus partiële aanpak: Bij de totaalaanpak wordt gestreefd naar een omvattend totaalsysteem. De partiële aanpak streeft naar het reduceren van de complexiteit door decompositie;
- lange versus korte levensduur van informatiesystemen.

De hierboven genoemde ontwerpfilosofieën en de in (Geurden, 1988), (Meyer, 1988) en (Tsichritzis, 1989) beschreven object-georiënteerd gerichte ontwerpfilosofieën hebben als uitgangspunt gediend bij het samenstellen van de criteria, waaraan de object-georiënteerde benadering getoetst zal worden.

3.2 Criteria

- Lengte ontwikkelperiode
  Door de bevriezingsperiode te beperken, hergebruik van software te bevorderen en een zodanige systeemstructuur te kiezen waar veranderingen makkelijk kunnen worden aangebracht, is het mogelijk de effectiviteit en de efficiency van de systeemontwikkeling te verhogen.

- De mogelijkheid voor modulaire decompositie door het probleem op te delen in hanteerbare subproblemen (Meyer, 1988), (Geurden, 1988).
  De indeling in subsystemen dient zodanig te geschieden dat deze beter past bij de veranderlijke werkelijkheid.

- De mogelijkheid tot beperking van interfaces (Meyer, 1988), (Geurden, 1988).
  Door het beperken van het aantal interfaces wordt de autonomiteit van clusters (modules) verhoogd. Deze beperking kan worden bereikt door de clusters (modules) binnen een systeem zwak te koppelen (Geurden, 1988). Daarentegen dienen de elementen waaruit een cluster (module) bestaat een sterke onderlinge samenhang (cohesie) te hebben. Dit vereist een hiërarchische structuur, waarbij hoger gelegen modules de onderliggende het gedetailleerde werk laten doen.

- De mogelijkheid tot hergebruik (Meyer, 1988), (Geurden, 1988), (Tsichritzis, 1989). Dit kan bereikt worden door eerder gedefinieerde delen samen te voegen tot een nieuw systeem. Deze benadering vereist een eenvoudige toegang tot aanwezige kant-en-klare-delen, hetgeen momenteel een groot probleem is bij systeemontwikkeling.

- Verbeteren van de communicatie.
  In het algemeen kan gesteld worden dat de communicatie tussen gebruiker en ontwikkelaar verbeterd dient te worden. Dit kan bereikt worden door een betere representatiewijze van de systeemontwikkelingsmodellen te hanteren, zodat deze beter aansluit bij de belevingswereld van de mensen en de modellen meer overeenstemming vertonen met de werkelijkheid.

- De mogelijkheid tot integratie.
  Door de toenemende eis tot systeemintegratie op zowel organisatorisch als bedrijfstakniveau en de toenemende complexiteit in de omgeving, dient de te hanteren benadering mogelijkheden te bieden om complexe situaties te analyseren en flexibele systemen te bouwen, waardoor de systemen als een geïntegreerd totaalsysteem gezien kunnen worden.

4 Toetsing

De object-georiënteerde benadering zal aan de hand van de in paragraaf drie opgestelde criteria worden getoetst.

4.1 Toetsing aan de criteria

- Lengte ontwikkelperiode.

De object-georiënteerde benadering biedt de mogelijkheid, om een informatiesysteem op te delen in modules, waarbij elke module bestaat uit een aparte datastructuur met daarbij alle processen die deze data direct bewerken. De modules onderling hoeven niet op de hoogte te zijn van elkaars inwendige structuur daar een module de data in een andere module alleen kan beïnvloeden via interfaces. Hierdoor kunnen de verschillende modules gelijktijdig ontwikkeld worden door kleine teams, waarbij eventuele veranderende gebruikerseisen direct kunnen worden aangepast in de betreffende module, hierdoor kan een systeem sneller en beter worden aangepast aan de constante veranderingsbehoeften. Bij de object-georiënteerde benadering worden de ontstane objectklassen in een objectklassen-bibliotheek opgeslagen. Deze objectklassen kunnen worden aangepast en uit bestaande objectklassen kunnen met behulp van object-georiënteerde talen nieuwe objectklassen worden gecreëerd. Door gebruik te maken van bestaande objectklassen kan de effectiviteit en efficiency van de systeemontwikkeling worden vergroot, waardoor de doorlooptijd verkort kan worden. In (Tsichritzis, 1989) wordt aangegeven dat hergebruik van software kan leiden tot zowel snellere als goedkopere systeemontwikkeling.

- De mogelijkheid voor modulaire decompositie.

Bij de object-georiënteerde benadering is het mogelijk in de analysefase het probleemgebied in logische objectklassen onder te verdelen. Iedere objectklasse of groep van objectklassen kan hierna als probleemgebied beschouwd en apart uitgewerkt worden. Door de data en de services (functionaliteiten) die deze data bewerken samen te voegen binnen een objectklasse zijn deze gemakkelijk te hanteren.

- De mogelijkheid van interface beperking.

- De mogelijkheid van interface beperking.

Het aantal interfaces bij de object-georiënteerde benadering wordt beperkt door een samenvoeging van data en bijbehorende services in een objectklasse. Hierdoor ontstaan bouwstenen (clusters, bestaande uit een of meerdere objectklassen) waarbij kennis van de inwendige structuur van elke bouwsteen niet van belang is voor de andere bouwstenen, waardoor een zwakke koppeling tussen de objectklassen onderling ontstaat. Binnen de inwendige structuur van een bouwsteen (cluster) is de koppeling sterk, waardoor er sprake is van een hoge cohesie binnen de bouwsteen (cluster). De inhoud van een bouwsteen (cluster) kan alleen via een interface benaderd worden. Deze interface bestaat uit messages (berichten), die van en naar de bouwstenen (clusters) worden verstuurd. De bouwsteen (cluster) waarnaar een message (bericht) wordt gestuurd, kan afhankelijk van diens interne staat, het soort message (bericht) dat wordt ontvangen en de aard van de eventuele argumenten, een bepaalde handeling verrichten. Een ring van messages (encapsulation) waarmee implementatiedetails worden verborgen, zorgt voor de benodigde information hiding (abstractie mechanisme), waardoor de interne staat effectief is beschermd.

- De mogelijkheid tot hergebruik.

Door vanaf de OOA en de OOD gebruik te maken van standaard objectklassen uit de objectklassen-bibliotheek wordt hergebruik op een hoog abstractie niveau bevorderd. Hierbij is het noodzakelijk dat de objectklassen over een goede toegankelijke beschrijving beschikken. Tevens dient een selectiemechanisme beschikbaar te zijn, dat de standaard objectklassen uit de objectklassen-bibliotheek selecteert. Object-georiënteerde talen bieden de mogelijkheid om door middel van inheritance (overerving) objectklassen kennis met elkaar te laten delen. Op deze manier kunnen nieuwe objectklassen gecreëerd worden als specialisatie (inheritance downward) of combinaties van andere objectklassen (inheritance upward). Door aanpassing van de services (functionaliteiten) is het mogelijk bestaande objectklassen opnieuw te gebruiken. De kwaliteit van herbruikbaarheid zal toenemen naarmate de verzameling van bestaande, algemeen bruikbare objectklassen wordt uitgebreid. Het inheritance-mechanisme (overerving) ondersteunt een manier van programmeren door verfijning waardoor redundantie vermeden wordt.

- Verbeteren van de communicatie.

Het uitgangspunt van de denkwijze bij object-georiënteerd is gebaseerd op het denken in objecten, in eigenschappen, kenmerken en gedrag van die objecten, in hele objecten en hun componenten en in klassen van objecten en hun leden. In (Coad, 1990) wordt aangegeven dat deze manier van denken goed aansluit bij de manier waarop mensen tegen de werkelijkheid aan kijken. Hierdoor worden betere kansen gecreëerd om een juiste afstemming tussen de wensen van gebruikers aan de ene kant en ideeën van ontwikkelaars aan de andere kant te krijgen, waardoor het communicatie probleem geminimaliseerd zou kunnen worden.

- De mogelijkheid tot integratie.

De ontwikkelingen op het gebied van de informatietechnologie hebben er toe geleid dat deze technologie binnen steeds meer gebieden kan worden toegepast, waardoor de complexiteit wordt verhoogd en de vraag naar flexibele en geïntegreerde systemen is toegenomen. Doordat de object-georiënteerde benadering uitgaat van clusters, waarbij de inwendige structuur van de cluster niet bekend hoeft te zijn voor andere clusters en communicatie uitsluitend geschiedt met behulp van messages (berichten) is het mogelijk om koppelingen via interfaces tot stand te brengen en het systeem uit te breiden, waardoor de integreerbaarheid en de flexibiliteit toeneemt.

5 De gevolgen voor de huidige systeem-ontwikkeling

De huidige denkwijze bij ontwikkelmethoden zal aangepast moeten worden om de in paragraaf 1.3 genoemde knelpunten op te lossen. Tevens zullen de ontwikkelfasen op een andere wijze benadrukt dienen te worden en zal de aanpak een iteratief karakter dienen te krijgen.

In eerste instantie dient de analysefase, gedurende welke de gebruikerseisen worden verzameld en vastgelegd, in een korte doorlooptijd te worden uitgevoerd, daar de gebruikersbehoeften aan sterke veranderingen onderhevig zijn (zie figuur 1.1). De manier waarop analisten analyses zullen gaan uitvoeren, zal gebaseerd dienen te zijn op het werken met bouwstenen, waarbij de inwendige structuur van de bouwsteen niet belangrijk is, maar wel de interactie tussen deze bouwstenen. De object-georiënteerde systeemontwikkeling vraagt van de analisten een denkwijze die gebaseerd is op het denken in objecten, in eigenschappen, kenmerken en gedrag van die objecten en in klassen van objecten en hun leden. Het uitgangspunt van de OOA dient het objectsysteemmodel (waarom) te zijn en de aspectmodellen die door het informatiesysteem wordt beïnvloed. De OOA-modellen bevatten objectklassen uit het probleemgebied, waarin het data- (statische structuur), het gedrags- (dynamische structuur) en het functionele aspect (statische structuur) van die objectklassen tot uiting komen. Hierdoor worden de objectklassen op een betere manier beschreven, waardoor bij een eventueel hergebruik een toegankelijke beschrijving aanwezig is.

In de ontwerpfase worden de objectklassen in clusters ingedeeld, zodat een module structuur ontstaat. Deze clusters worden ieder afzonderlijk en eventueel parallel aan elkaar uitgewerkt en geïmplementeerd. Deze werkwijze kan gehanteerd worden daar de inwendige structuur van een cluster voor een andere cluster niet bekend hoeft te zijn. Bij de verdere invulling van de clusters zullen bestaande objectklassen centraal moeten worden gesteld in plaats van te werken van globaal naar detail en van vaag naar minder vaag. Dit vereist de aanwezigheid van een goede objectklassen-bibliotheek, die beheerst dient te worden. Tevens dient een selectie-mechanisme aanwezig te zijn, waarmee de relevante objectklassen geselecteerd kunnen worden.

In de realisatiefase zal de nadruk komen te liggen op het hergebruik van programmacode door het creëren van nieuwe objectklassen uit bestaande objectklassen. Om dit te kunnen bewerkstelligen dienen OO-talen de volgende basisprincipes te ondersteunen: encapsulation (inkapseling), information hiding, inheritance (overerving) en polymorfisme.

Het onderhoud krijgt een geheel ander karakter doordat de nadruk minder komt te liggen op het oplossen van fouten in programmacode, maar op het

onderhoud bestaat voornamelijk uit het aanpassen en verbeteren van de methods (acties) en het uitbreiden van de services (functionaliteiten) van bestaande objectklassen.

De organisatiemodellen, de hieruit afgeleide objectsysteemmodellen en de daaruit afgeleide informatiesysteemmodellen (zie figuur 2.2) liggen ingekapseld opgeslagen in de object database, waardoor het integratie-aspect wordt verhoogd.

Voor wat betreft de object-georiënteerde systeemontwikkeling kan gezegd worden dat nog veel werk en onderzoek zal moeten worden verricht om een ontwikkelmethodiek op te stellen die het integratie-aspect beschreven in paragraaf 2.2. ondersteunt en waarin kenmerken van de denk- en werkwijze (paragraaf 2.2.) tot uiting komen. De meningen lopen sterk uiteen omtrent de vraag of de huidige systeemontwikkelmethoden en -technieken gebruikt kunnen worden in combinatie met een geheel of gedeeltelijk object-georiënteerde benadering. Hier zal nog veel onderzoek naar moeten worden verricht.

Object-georiënteerde systeemontwikkeling zal gedurende de negentiger jaren meer gestalte gaan krijgen, waarbij een integratie van de verschillende invalshoeken (zie figuur 2.2) dient plaats te vinden zodat vanuit eenzelfde basisfilosofie vertaald in modelleringsprincipes het conceptualiseren (visualiseren), het transformeren en implementeren wordt uitgevoerd (zie figuur 2.1).

Duidelijk is dat de object-georiënteerde benadering een steeds belangrijker rol inneemt en dat nog vele overgangsfasen zullen volgen om uiteindelijk tot een volledige object-georiënteerde systeemontwikkelings-omgeving te komen.

Literatuur

ABBENHARDT, H.
Datenbanken für software-entwicklungsumgebungen-techniken im vergeleich,Information Management, 1990.

BARTELS-MERTENS, A.C.M., A.J.M. BEULENS, C.A. VAN LOON, D. SCHUYL-LAROS
Object-georiënteerde systeemontwikkeling een trend of toekomst?, IT-Report Series nr. 1, Den Haag, Haagse Hogeschool, sector Informatica, 1991.

BEMELMANS, TH.M.A.
Ontwikkelingsmethoden voor informatiesystemen: onopgeloste problemen, NGI-bundel 'Methodieken voor informatie-systeemontwikkeling', 1986.

BEMELMANS, TH.M.A.,
Bestuurlijke informatiesystemen en automatisering, derde herziene druk, Leiden, Stenfert Kroese, 1987.

BOOCH, G.,
Object oriented design, with applications, z.dr., Menlo Park, The Benjamin/ Cummings Publishing Company, z.j..

COAD, P., E. YOURDON,
Object-oriented analysis, z.dr., Englewood Cliffs, NJ, Prentice Hall, 1990.

GALBRAITH, J.R.,
Het ontwerpen van complexe organisaties, z.dr., Alphen aan de Rijn, Samson Uitgeverij, 1976.

GEURDEN, W.,
Object oriented programming, Beleidsinformatica Tijdschrift, 1988.

GUPTA, Y.P.,
Directions on structured approaches in system development, IMDS, Juli/Augustus 1988.

HENDERSON-SELLERS, B., J.M. EDWARDS,
Object-oriented systems life-cycle, Communications of the ACM 33, 9 (September 1990), 143-159.

JACOBSON, I.,
Object oriented development in an industrial environment, OOPSLA '87 Proceedings, 1987

KATWIJK, J. VAN,
Object-georiënteerde software engineering, Database Magazine, 1990.

MEYER, B.,
Object oriented software construction, z.dr., Englewood Cliffs, NJ, Prentice Hall, 1988.

MINTZBERG, H.,
The structuring of organizations, z.dr., Englewood Cliffs, NJ, Prentice Hall, 1979.

PAGE-JONES, M.,
object-georiënteerd werken kan goed, mits rekening wordt gehouden met het verleden, Computerworld, September 1990.

RADEMACKERS, G.J.,
Wat is object-oriented programmmeren, Journal of Software Research, april 1989.

SHLAER, S., S.J. MELLOR,
Object oriented systems analysis: modeling the world in data, z.dr., Englewood Cliffs, NJ, Prentice Hall, 1988.

SMITH, M.K., S.R. TOCKEY,
An integrated approach to software requirements definition using objects, z.dr., z.p., Boeing Commercial Airplane - Support Division, Boeing Computer Services, z.j.., 1988

SUNDGREN, B.,
Een raamwerk voor het ontwerpen van informatiesystemen, Informatie 20, 1 (1978).

TSICHRITZIS, D.,
Object oriented development for open systems, Proceedings of IFIP, 1989.

WAES, R.M.C.,
Architectures for information management, a pragmatic approach on Architectural Concepts and their Application in Dynamic Environments, z.dr., z.p., Thesis Publishers, Amsterdam, 1991.

WELKE, R.J.,
Current information, systems analysis and design approaches, in education of large information systems, North Holland Publ. Co., Amsterdam, 1977.