

# **De metafysica van object georiënteerd programmeren**

Leringen van Plato en Aristoteles

Door : Kevin Breed  
Cursus : Filosofie van media en communicatie  
Werkgroep : 6  
Werkgroepdocent : Niels Helsloot  
Datum : 26 juni 2008

# Inhoud

Proloog .....	3
Inleiding: Object georiënteerd programmeren en Plato .....	4
De veronderstelling over Plato's ideeënleer .....	6
De veronderstelling .....	6
Samenvatting .....	8
Het concept van object georiënteerd programmeren en de overeenkomst met Plato's ideeënleer..	9
Wat is object georiënteerd programmeren?.....	9
Object georiënteerde programma's .....	10
Samenvatting .....	10
Object georiënteerd programmeren en Aristoteles logica .....	12
De essentie in de dingen .....	12
Samenvatting .....	14
De wisselwerking tussen het Aristoteliaanse en Platoonse perspectief.....	15
De perspectieven .....	15
De wisselwerking .....	16
Samenvatting .....	17
Conclusie .....	18

## Proloog

Dit paper komt voort uit de observatie van een aantal overeenkomsten tussen mijn kennis van object georiënteerd programmeren en filosofische inzichten. Als webdeveloper heb ik vooral ervaring met een aantal programmeertalen voor het World Wide Web, zoals PHP, Javascript en Actionsript. In het gebruik van deze talen heb ik mijzelf aangeleerd om object georiënteerd, in plaats van procedureel te programmeren. Door het volgen van de premaster Nieuwe Media en Digitale Cultuur aan de Universiteit Utrecht ben ik, onder andere door het vak 'Filosofie van media en communicatie', in aanraking gekomen met een aantal filosofische inzichten. Deze lijken soms vrij letterlijk terug te vinden in bepaalde elementen van object georiënteerd programmeren. Omdat deze verbanden naar mijn weten nog niet beschreven zijn heb ik met dit paper onderzocht in hoeverre de overeenkomsten tussen object georiënteerd programmeren en filosofie te maken zijn. Dit paper is daarom vooral een exploratie van de raakvlakken tussen deze twee domein en de manier waarop object georiënteerd programmeren in een filosofisch perspectief begrepen kan worden. Omdat de filosofie een erg breed veld is, heb ik mij beperkt tot de metaphysica van Plato en Aristoteles. Alhoewel lichtelijk gedateerd zijn hun opvattingen vandaag de dag nog steeds in meerdere opzichten aanwezig. Ook hun aanwezigheid in object georiënteerd programmeren kan op meerdere manieren belicht worden. De centrale onderzoeksvraag luidt dan ook: "Welke vooronderstellingen zijn er in object georiënteerd programmeren te onderscheiden wanneer het perspectief van Plato's ideeënleer en Aristoteles logica wordt toegepast?" Met dit onderzoek hoop ik andere inzichten in object georiënteerd programmeren verkregen te hebben die verder kijken dan het instrumentele perspectief waarin het meestal bekeken wordt.

## Inleiding: Object georiënteerd programmeren en Plato

Object georiënteerd programmeren heeft in metafysisch opzicht een duidelijke overeenkomst met Plato's ideeënleer. Ik ben dan ook niet de enige, of eerste die dit opmerkt. Wanneer Google de zoekterm "object oriented programming plato" verwerkt, worden er tijdens de tijd van schrijven 66.800 resultaten gevonden. Onder de (eerste) resultaten worden niet alleen de woorden uit de zoekterm toevallig op dezelfde pagina aangetroffen, maar zijn er ook een aantal aardige discussies te vinden over object georiënteerd programmeren en Plato (bijv. Tarko, 2006; Salas, 2005; Cyocum, 2004). Ook zijn er enkele academische stukken te vinden over dit onderwerp, zoals die van Ray Giguette (2006) en gedeeltelijk in Derek Rayside & Gerard Campbell (2000). Maar de overeenkomst is waarschijnlijk het meest direct bij Alan Kay, één van de ontwikkelaars van één van de eerste object georiënteerde talen "Smalltalk". In zijn beschrijving over de geschiedenis van Smalltalk merkt hij op:

"Its way of making objects is quite Platonic in that some of them act as idealisations of concepts--Ideas--from which manifestations can be created. That the Ideas are themselves manifestations (of the Idea-Idea) and that the Idea-Idea is a-kind-of Manifestation-Idea-- which is a-kind-of itself, so that the system is completely self-describing-- would have been appreciated by Plato as an extremely practical joke [Plato]." (1993)

Dat de overeenkomst door Plato gewaardeerd zou worden als een "extremely practical joke", geeft gelijk aan dat er allerm minst sprake is van een evenredige vergelijking. Zoals Rayside & Campbell opmerken dwingt de activiteit van het programmeren de programmeur niet om een metafysische positie in te nemen over de aard van de werkelijkheid (2000). Terwijl object georiënteerd programmeren een poging is om de alsmaar complexere dynamische systemen en gebruikersrelaties efficiënter te modelleren (Kay, 1993), is Plato's ideeënleer een oplossing voor het metafysische probleem van universaliteit en constante verandering. Deze vergelijking heeft dan ook niet als doel om een antwoord te geven of Plato's ideeënleer en object georiënteerd programmeren hetzelfde zijn. Het is een exploratie van de raakvlakken en de manier waarop deze raakvlakken inzichten kunnen geven in de manier waarop object georiënteerd programmeren begrepen kan worden.

Deze exploratie wordt gestart vanuit de veronderstelling waarmee object georiënteerd programmeren en Plato's ideeënleer overeen komen, zoals deze is terug te vinden in de

internetdiscussies, bij Rayside & Campbell en in bovenstaande quote van Kay. In hoofdstuk één zal ik beargumenteren dat de opvatting niet geheel juist is. In hoofdstuk twee is het concept van object georiënteerd programmeren uiteengezet en verbonden met de bevindingen uit hoofdstuk één. Hieruit is gebleken dat object georiënteerd programmeren beter vanuit een Aristoteliaans perspectief bekeken kan worden. Alhoewel Rayside & Campbell uitvoerig hebben beschreven hoe Aristoteliaanse logica overeenkomt met object georiënteerd programmeren, hechten zij niet veel waarde aan het verschil in opvattingen tussen Plato en Aristoteles. Nadat in hoofdstuk drie aandacht besteedt is aan de overeenkomsten tussen Aristoteles en object georiënteerd programmeren zal in hoofdstuk vier aandacht besteedt worden aan de wisselwerking tussen een Platoons en Aristoteliaans perspectief op object georiënteerd programmeren. Tot slot zijn in hoofdstuk vijf de bevindingen uit de voorgaande hoofdstukken globaal teruggekoppeld naar de stromingen van het modernisme en postmodernisme.

## **De veronderstelling over Plato's ideeënleer**

In dit hoofdstuk zal onderzocht worden of de veronderstelling die gemaakt wordt over Plato's ideeënleer klopt. In de internetdiscussies, de tekst van Kay (1993) en de tekst van Rayside en Campbell (2000) is deze als volgt:

1. In Plato's ideeënleer wordt er een onderscheid gemaakt tussen de Idee en instanties van de Idee.
2. De Idee kan beschouwd worden als een blauwdruk waaruit objecten geïnstantieerd worden.

Bij het onderzoek naar deze veronderstelling zal aandacht worden besteedt aan wat Plato onder de Idee en instanties van de Idee verstaat en hoe deze zich tot elkaar verhouden. Hieruit zal blijken dat de veronderstelling in grote lijnen klopt, maar er een aantal cruciale verschillen op te merken zijn.

### ***De veronderstelling***

*1. In Plato's ideeënleer wordt er een onderscheid gemaakt tussen de Idee en instanties van de Idee.*

Deze opvatting is waar, maar misleidend. Plato's Ideeënwereld bestaat uit pure simpele Ideeën, zoals Schoonheid, Moed en Verstand, maar ook uit aardse zoals Appelheid, Tafelheid en Schoenheid. Deze wereld verschilt van de wereld zoals deze aan ons verschijnt en waarin we leven. Deze bestaat uit specifieke dingen, die ik vanaf nu individuen zal noemen. Individen zijn samengesteld uit en onder te verdelen in de Ideeën. Echter zijn deze individuen altijd imperfect, omdat ze nooit een volledige en pure instantie zijn van de Idee. In plaats van een 'afgeleid van', zoals instantie hier bedoeld wordt, spreekt Plato van een 'delen in' om de relatie tussen de individuen en Ideeën te beschrijven (Silverman, 2003). De individuen delen in meerdere Ideeën om zich aan ons te laten verschijnen, waardoor ze complexer worden dan de Idee zelf. De individuen zijn dus geen exacte afgeleiden (instanties) van een blauwdruk, maar een veranderende samenvoeging van meerdere Ideeën. Dit maakt de bovenstaande bewering misleidend, omdat het woord een ultiem model suggereert waaruit de werkelijke dingen om ons heen afgeleid worden. Wanneer we echter uitgaan van een 'delen in' dan zijn de individuen geen

substracties van een ultiem model, maar een verschijning dat in de Idee deelt, maar het niet ten volste is. Het enige wat de Idee ten volste is, is de Idee zelf. Deze relatie noemt Plato 'zijn'. De idee deelt niet in zichzelf, maar Is simpelweg zichzelf. De wereld van individuen, de wereld waarin we leven, is volgens Plato daarom slechts een schijnwereld. Alles *deelt* in Ideeën, maar niets *Is* de Idee.

*2. De Idee kan beschouwd worden als een blauwdruk waaruit objecten geïnstantieerd kunnen worden.*

Nu we hebben gezien dat de individuen niet simpelweg begrepen kunnen worden als exacte afgeleiden (instanties), is de bewering dat de Idee als blauwdruk beschouwd kan worden ook onwaarschijnlijker. Echter levert dit een moeilijkheid op met Plato's zijn opvatting dat de fysieke wereld en alle dingen in de wereld, kopieën, of afbeeldingen zijn van de Ideeën, waardoor de fysieke wereld afhankelijk wordt van de Ideeënwereld (Silverman, 2003). Wanneer de samenstelling van Ideeën als eigenschappen van het individu beschouwd worden, dan kan het individu zelf geen Idee zijn, of een afgeleide van een Idee, omdat het meer is dan zichzelf. Een Idee is immers alleen zichzelf en niets anders dan zichzelf. Maar dit neemt niet weg dat de eigenschappen van het individu geen afgeleiden van Ideeën zouden kunnen zijn. Dan zouden de Ideeën wel als blauwdruk beschouwd kunnen worden en zouden hun instanties zich manifesteren in de eigenschappen van het individu. De eerste veronderstelling zou dan ook kloppen, alleen zouden instanties beschouwd moeten worden als eigenschappen van een individu en niet als iets dat op zichzelf kan bestaan. De manier waarop Plato de fysieke wereld als kopie, of afbeeldingen beschouwd is dan echter geen één op één relatie. De fysieke wereld is geen afgietsel van de Ideeënwereld, maar samengesteld uit kopieën van de Ideeënwereld. Dit zou suggereren dat we de Idee ook in deze wereld kunnen vinden in de eigenschappen van het individu. Dit is echter problematisch, om twee redenen. Ten eerste neemt de kopie van de Idee deel in de individu en kan als zodanig niet op zichzelf worden beschouwd. Wanneer de Idee van Schoonheid in Jessica deel neemt kunnen we zeggen dat Jessica mooi is. Dit wil niet zeggen dat Jessica zelf Schoonheid is, zij bezit alleen deze eigenschap. We weten nu nog niets over de Idee van Schoonheid zelf. Ten tweede is het bezitten van deze eigenschap onderhevig aan verandering. De Schoonheid in Jessica is niet onveranderlijk. Het is een eigenschap die kan veranderen naarmate de tijd verstrijkt. Voor Plato is het dus problematisch om kennis op te doen uit de eigenschappen van individuen, ofwel uit de kopie van de Idee. Dit probleem is afkomstig van Heraclitus, een Griekse filosoof voor Plato. Hij stelde dat de wereld om ons heen constant verandert waardoor het opdoen van

universele, absolute kennis onmogelijk wordt. Iets waar we het ene moment kennis van nemen kan het volgende moment anders zijn, waardoor de opgedane kennis niet langer meer geldig is. Heraclitus nam dit gegeven zoals het was en beruste in de notie dat verandering het enige is waar we zeker van kunnen zijn. Plato ruste hier echter niet in. Hij kon zien dat de wereld om ons heen constant verandert, maar hij geloofde ook dat er zoiets bestond als universele absolute kennis. Wanneer de kopie van de Idee van Schoonheid in Jessica verandert, en misschien zelfs helemaal verdwijnt, betekent dit dan dat de Idee Schoonheid niet bestaat? In dit voorbeeld is deze constatering nog onproblematisch, omdat andere individuen de Idee van Schoonheid ook kunnen bezitten. Stel echter dat de zon zou doven en we bij de gratie van een licht zo zwak als de maan zouden moeten leven, dan zouden we kleuren zoals het groen van een blad van een tulp niet meer kunnen waarnemen. Betekent dit dat de Idee Groenheid niet meer bestaat? Volgens Plato niet. De Idee Groenheid zou zich weliswaar hebben teruggetrokken uit het individu, maar zou los van tijd en ruimte blijven bestaan in de Ideeënwereld. Dit gaf Plato de mogelijkheid om een universeel en absoluut bestaansrecht aan kennis te geven, waardoor we niet gebonden zijn aan de onvermijdelijkheid van Heraclitus verandering.

## ***Samenvatting***

In dit hoofdstuk is de veronderstelling onderzocht dat (1) In Plato's ideeënleer een onderscheid wordt gemaakt tussen de Idee en instanties van de Idee. (2) De idee beschouwd kan worden als blauwdruk waaruit objecten geïnstantieerd worden. Deze opvatting ligt als zodanig ten grondslag aan de overeenkomst tussen Plato's ideeënleer en object georiënteerd programmeren. Alhoewel de opvatting in grote lijnen klopt zijn er een aantal cruciale verschillen op te merken. De tweede stelling is onjuist, omdat uit Plato's ideeën geen objecten (of individuen) geïnstantieerd kunnen worden, maar slechts eigenschappen van objecten. De eerste stelling wordt daardoor misleidend. Alhoewel er een onderscheid gemaakt kan worden tussen de Idee en instanties van de Idee, kan de Idee niet gezien worden als ultiem model voor de individuen, of de fysieke wereld. Wel is het een ultiem model voor de eigenschappen van de individuen, ofwel de instanties, of kopie-Ideeën van de pure Idee. De belangrijkste lering van Plato is dat pure Ideeën niet gevonden kunnen worden in de wereld om ons heen, of in de eigenschappen van individuen, omdat de wereld constant verandert.

## **Het concept van object georiënteerd programmeren en de overeenkomst met Plato's ideeënleer**

In het vorige hoofdstuk is gebleken dat de veronderstelling over Plato's ideeënleer gecompliceerder is dan dat deze in eerste instantie lijkt. Hierdoor kan ook de overeenkomst met object georiënteerd programmeren minder gemakkelijk gemaakt worden. In dit hoofdstuk zal aandacht geschonken worden aan het concept van object georiënteerd programmeren en de overgebleven raakvlakken met Plato's ideeënleer.

### ***Wat is object georiënteerd programmeren?***

Object georiënteerd programmeren is simpelweg een programmeer paradigma. Door een aantal specifieke syntactische mogelijkheden in een programmeertaal wordt het mogelijk om klassen te schrijven waaruit objecten geïnstantieerd kunnen worden. Klassen zijn onder te verdelen in eigenschappen en methodes. Eigenlijk komt dit erop neer dat bijvoorbeeld de klasse "Appel" de eigenschap "kleur" kan hebben, waarvan de inhoud variabel is, zoals "rood", "groen", of "paars". Methodes van een klasse verzorgen een bepaalde functionaliteit. Zo zou de "Appel" klasse de methode "groeien" kunnen hebben, waardoor de grootte van de appel over een bepaald tijdsbestek zal toenemen. Object georiënteerd programmeren maakt het mogelijk om de functionaliteit die voor een applicatie nodig is onder te verdelen in een object model. Anders dan bij procedureel programmeren heeft dit tot gevolg dat het programma opgedeeld wordt in zelfstandig naamwoorden, in plaats van werkwoorden. Zo zou een appelboomgaard applicatie onderverdeeld worden in appelbomen, bomen, taken, bladeren, appels, etc., terwijl met procedureel programmeren de appelboomgaard in functies onderverdeeld wordt, zoals groeien, rijpen, plukken, etc. Deze onderverdeling wordt dus gemaakt aan de hand van de te onderscheiden 'objecten', welke in klassen gedefinieerd worden, door middel van eigenschappen en methodes.

Object georiënteerd programmeren kan op twee manier gezien worden, als activiteit en als software. In het laatste geval gaat het dan niet meer over programmeren, maar over een object georiënteerd programma. In het eerste geval betreft het niet alleen het schrijven van de code, maar ook de manier waarop het probleem begrepen wordt. Zowel object georiënteerd programmeren, als object georiënteerde programma's vallen onder het object georiënteerde concept.

## ***Object georiënteerde programma's***

Net zoals Plato het onderscheid maakt tussen de Idee en instanties van de Idee, kan bij object georiënteerde programma's een onderscheid gemaakt worden tussen de klasse en de instantie van de klasse, die het object genoemd wordt. Bij Plato's ideeënleer hebben we echter geconstateerd dat een instantie, ofwel een kopie van de Idee, alleen in een eigenschap van een individu, dat we als object zouden kunnen zien, zou kunnen bestaan. In een object georiënteerd programma bestaan instanties van objecten echter als individu waarvan de samenstelling van eigenschappen en methodes in de klasse gedefinieerd is. Alhoewel er op dit gebied geen precieze overeenkomst is, is er misschien wel een overeenkomst op het gebied van abstractie. Object georiënteerd programmeren had immers tot doel om de complexiteit van dynamische systemen te verminderen, of op zijn minst beter begrijpbaar te maken. Door de controlestructuren waaruit een programma is opgebouwd te ordenen in de eigenschappen en methodes van klassen, wordt er niet alleen een taxonomie aangebracht, maar wordt er ook een stukje complexiteit gedelegeerd, of in andere woorden, geblackboxd. Een object, ofwel een instantie van een klasse is tijdens de uitvoer van het programma verantwoordelijk voor zijn eigen functionaliteit. Een object is daarmee minder complex dan de klasse waaruit hij afgeleid is, waardoor er een simpeler idee over zijn wezen opgedaan kan worden. Deze abstractie werkt precies andersom als bij Plato's Ideeënleer, waar de individuen (objecten) complexe samenstellingen zijn van monistische Ideeën.

Het is belangrijk om op te merken dat de eigenschappen en methodes van een klasse, of object, anders zijn dan Platoonse eigenschappen. Platoonse eigenschappen zijn kopieën van Ideeën die buiten de fysieke wereld bestaan. Een eigenschap van een object kan in een object georiënteerd programma echter uit een ander object bestaan. Alhoewel deze eigenschap in eerste instantie kan leiden tot een simpel idee dat het object in de eigenschap door zijn delegatie oproept, krijgt dat object zijn identiteit door zijn eigen eigenschappen en methodes. Dit sluit beter aan bij het Aristoteliaanse perspectief dat ik in het volgende hoofdstuk uiteen zal zetten.

## ***Samenvatting***

In dit hoofdstuk heb ik het object georiënteerde concept uiteengezet en aandacht geschonken aan object georiënteerde programma's. De objecten uit deze programma's zijn minder complex dan de klassen waaruit ze afgeleid worden. Hierdoor is de relatie tussen klasse en object het omgekeerde als de relatie tussen Plato's Idee en instanties. Beiden leiden ze echter tot een abstractie van complexiteit. De manier waarop deze abstractie zich in de eigenschappen van een

object manifesteert sluit echter beter aan bij een Aristoteliaans perspectief dat in het volgende hoofdstuk uiteengezet zal worden.

## Object georiënteerd programmeren en Aristoteles logica

In het vorige hoofdstuk is het concept van object georiënteerdheid uiteengezet. Dit viel uiteen in object georiënteerd programmeren en object georiënteerde programma's. Dit laatste element is in het vorige hoofdstuk uiteengezet en in verband gebracht met Plato's ideeënleer. Daaruit bleek dat de manier waarop objecten eigenschappen kregen beter aansluit bij een Aristotelianse logica die in dit hoofdstuk nader toegelicht zal worden. In dit hoofdstuk zal ook aandacht worden besteedt aan het object georiënteerde concept als programmeren.

### ***De essentie in de dingen***

Als de leerling van Plato was Aristoteles het niet geheel eens met Plato's ideeënleer. Hij vond wel dat er een universele, absolute, onveranderlijke waarheid bestond, maar zocht deze in de dingen in de wereld om ons heen. Hij dacht niet dat er een hogere orde van pure Ideeën bestond, maar vond met de instrumenten der logica een universele essentie in de dingen zelf. Deze universele essentie bestond volgens Aristoteles alleen als abstractie in de geest van de mensen en niet als entiteit in een andere wereld. Rayside & Campbell beargumenteren dat object georiënteerd programmeren Aristotelian is wanneer de programmeur eerst kennis opdoet over individuen (objecten) in het probleemdomen en daarna abstracties ontwikkelt en definieert in een klasse (2000). Deze activiteit heeft betrekking op het object georiënteerde concept van programmeren. Hierbij gaat het in eerste instantie over de activiteit om een probleem te inventariseren en om te zetten naar een oplossing. Het schrijven van de klasse zelf blijft in eerste instantie nog buiten beschouwing alsmede het object georiënteerde programma.

Terwijl Plato een hogere ontologische status aan de Ideeën toekent, kent Aristoteles een hogere status toe aan de wereld om ons heen. De dingen zelf zijn de realiteit volgens Aristoteles en het is niet nodig om deze als schimmen in een schijnwereld af te doen. Voor Aristoteles zijn er geen schimmen, maar dingen die gedefinieerd en van elkaar gedifferentieerd kunnen worden door essentiële en toevallige onderscheiden. Dit begint met het definiëren, ofwel een eenheid in de dingen aanbrengen. Deze eenheden worden 'soorten' genoemd, waarbij iets enkels gezegd kan worden over vele individuen. Zo kan over zowel Jan, Piet als Clara gezegd worden dat ze mens zijn. Dit vertelt ons iets over het soort van de individuen. Het is een universele abstractie die los van de veranderende individuen bestaat. Wanneer soorten verder geabstraheerd worden, dan wordt deze onderverdeling *genus* genoemd. Dit is wanneer iets enkels gezegd kan worden over vele soorten. De mens valt bijvoorbeeld onder de *genus* 'dier'. Deze abstracties van soort en *genus* hebben allemaal eigenschappen, waarvan sommige toevallig en andere essentieel zijn.

Waar het bij Aristoteles op neer komt is dat om te kunnen weten wat voor soort een individu is, we het moeten onderscheiden van de andere soorten binnen de genus. Wanneer de eigenschappen van een soort onderverdeeld worden in essenties en toevalligheden komen we de ware aard van de soort te weten. De paradox in Aristoteles logica is dat we door deze onderverdelingen binnen de terminologie van abstracties blijft hangen. Maar wat komen we met deze taxonomieën te weten over wat iets is buiten de onderverdeling van essentiële en toevallige eigenschappen? Wanneer we zouden zeggen dat een plant niet alleen een plant, maar ook een organisme is (wat Aristoteles doet) dan hebben we twee ideeën over wat een plant is, die allebei iets zeggen over wat een plant is. De plant als organisme vertelt ons niet alleen minder over het zijn van de plant, het geeft ons ook een ander idee van wat de plant is. Het laat specifieke eigenschappen buiten beschouwing die bijdragen aan de identiteit van het plant zijn. Logisch gezien is de opvatting ‘een plant is een organisme’ waar. Tegelijkertijd is het ook niet waar, omdat een plant alleen een plant kan zijn en niets anders dan een plant. De abstractie leidt tot reductie.

Zowel Plato als Aristoteles zien individuen als composities. De abstracties zijn voor Plato echter een puur Idee, terwijl deze voor Aristoteles ook composities zijn. Hierdoor roept het antwoord op een vraag altijd een nieuwe vraag op. Wanneer we vragen ‘wat is groen?’, dan kunnen we antwoorden dat groen een kleur is. Als we vragen ‘wat is kleur?’, dan kunnen we antwoorden dat het een bepaalde breking van licht is dat op onsnetvlies valt en wordt omgezet naar een elektrisch seintje in de hersenen. Dan komen we bij de vragen ‘wat is licht?’, ‘wat is ons netvlies?’, ‘wat is elektriciteit?’, ‘wat zijn onze hersenen?’. Dit is het principe waar de natuurkunde op gestoeld is. We begrijpen de dingen om ons heen als samenstellingen van andere dingen, als complexe composities bestaande uit abstracties die zijn samengesteld uit andere abstracties die allemaal ook zijn samengesteld uit abstracties, etc. Leven we hierdoor niet juist in een schijnwereld waar niet is wat het lijkt? Deze schijn rust echter op een ander soort verandering dan waar Plato zijn Ideeënwereld een oplossing voor is. Het is geen verandering van de individuen, of eigenschappen van de individuen, maar een verandering in de manier waarop de Idee deelt in de individu.

De Aristoteliaanse universaliteit van de dingen is gelegen in de manier van samenstelling, ofwel hun structuur. In de manier waarop de allerkleinste deeltjes iets groters vormen, dat we als één begrijpen, maar eigenlijk uit velen bestaat. In deze nieuwe éénheden vormt het als combinaties, als structuur weer nieuwe eenheden, etc. De programmeur doet door het schrijven van klassen precies hetzelfde. Het ordent samenstellingen onder samenstellingen, waarbij iedere eenheid in de samenstelling zijn eigen structuur bepaald. Een klasse is dus altijd iets anders dan het is.

## ***Samenvatting***

In dit hoofdstuk is uiteengezet hoe Aristoteles door middel van logica een universele essentie uit de dingen zelf afleidt. In plaats van dat dit werkelijk iets zegt over een soort of genus is, leidt het eerder tot een universele taxonomie. Hierdoor wordt het mogelijk om uit alles een universeel model af te leiden, waarvan de keerzijde is dat de dingen niet zijn wat ze zijn, maar zijn waaruit ze essentieel of toevallig bestaan. Ditzelfde principe geldt voor de klassen van object georiënteerd programmeren, die ook uit eigenschappen en objecten bestaan die wellicht tot een soort behoren, maar ook eindeloos uit een samenstelling van andere eigenschappen bestaan.

## **De wisselwerking tussen het Aristoteliaanse en Platoonse perspectief**

In het vorige hoofdstuk is uiteengezet hoe de abstractie van Aristoteliaanse leidt tot reductie, waardoor het ding, of de soort altijd iets anders wordt dan zichzelf. Dit principe is ook aanwezig in het object georiënteerde concept. In dit hoofdstuk zal uiteengezet worden hoe zowel het Platoonse perspectief als het Aristoteliaanse perspectief met elkaar in wisselwerking zijn tijdens object georiënteerd programmeren. Hiervoor wordt eerst nog een keer uiteengezet hoe de perspectieven beschouwd kunnen worden, voordat de wisselwerking aan de hand van een voorbeeld uiteengezet wordt.

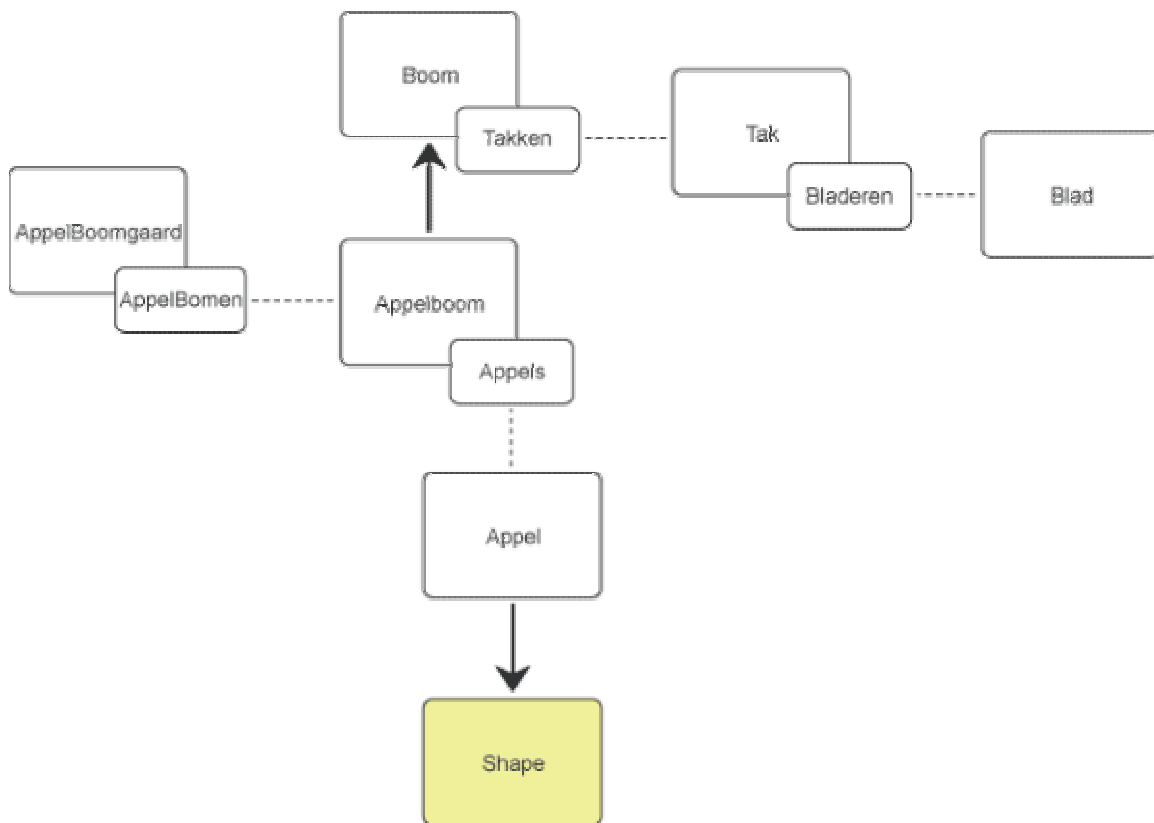
### ***De perspectieven***

We hebben nu een aantal dingen kunnen constateren. Bij object georiënteerd is er sprake van een probleemdomein. Dit probleemdomein is (een deel van) de wereld om ons heen zoals we deze waarnemen. Dit wordt door Plato als een schijnwereld gezien, maar door Aristoteles als de realiteit. Plato ziet de dingen om ons heen als schimmen die een complexe samenstelling van imperfecte instanties van pure Ideeën zijn. Wanneer we onze ziel losmaken van ons lichaam kunnen we in deze pure Ideeënwereld komen. Wanneer we een klasse als één puur Idee zouden beschouwen, los van zijn materiële drager en los van zijn eigenschappen (dat de klasse dus geheel ontdoet van zijn aard als container voor, of drager van iets anders) zouden we kunnen spreken van een Platoonse klasse, of van een Platoons object. Dit Idee bestaat al in de gedachte van de programmeur voordat de klasse een materiële substantie aanneemt. Dit Idee moet dan wel verheven zijn boven de schijnwereld en kan niet gezien worden als een object met eigenschappen. Wanneer het probleem domein gezien wordt als de realiteit, waaruit door middel van een Aristoteliaanse reductie een opdeling wordt gemaakt in verschillende klassen (objecten) welke bestaan uit een combinatie van andere objecten, die vorm krijgen door controle structuren, etc. kan er gesproken worden over een Aristoteliaanse klasse. Dit leidt tot een modellering van het probleemdomein. Dit maakt van een Aristoteliaanse klasse een blauwdruk voor zijn representatie. Wanneer het programma uitgevoerd wordt, wordt de klasse geïnstantieerd, waardoor het een individu wordt. De essentie van deze individu is echter nog steeds de abstractie die in een klasse, in het model is aangebracht. Deze instantie is voor Plato als representatie van de schijnwereld een kopie van een kopie en een illusie van een illusie, waardoor het verder van de pure Idee komt af

te staan. Een Aristoteliaanse instantie is dus in staat om een universele waarheid te bevatten, terwijl een Platoonse instantie slechts een illusie teweeg brengt.

## ***De wisselwerking***

Het maakt nogal een verschil welk perspectief toegepast wordt, maar eigenlijk zijn beide gelijktijdig aanwezig waardoor er een interessante wisselwerking ontstaat. Om deze duidelijk te maken wil ik gebruik maken van een voorbeeld. Stel dat het probleemdomen een appelboomgaard is. Dit probleemdomen, ofwel de realiteit, of schijnwereld is gegeven door een ruimdenkende fruitboer die een elektronisch schilderij aan zijn muur wil hebben met daarop een soort simulatie van een appelboomgaard. De programmeur gaat aan de slag en herkent duidelijk de objecten in de boomgaard en gaat ze onderverdelen. De programmeur begint met de klasse “AppelBoomgaard” welke hij een verzameling “AppelBomen” meegeeft als eigenschap. Een appelboom is natuurlijk een soort van boom. Daarom maakt hij de abstracte klasse “Boom”. Deze heeft de verzameling “Takken” die instanties van het “Tak” object bevat, deze instantie bevatten “Bladeren”, etc. (zie figuur 1).



*Figuur 1 – Object georiënteerd klassemodel van een AppelBoomgaard simulatie*

Objecten worden dus duidelijk gedefinieerd door hun eigenschappen en er is sprake van een Aristoteliaanse reductie. Dit model is afkomstig uit de realiteit en bevat dezelfde Aristoteliaanse essentie. In eerste instantie lijkt dit een stap in de goede richting voor de programmeur, maar wanneer hij de klassen uit gaat werken blijkt dat het conceptuele model uit het probleemdomen nog ernstig tekort schiet om door de computer uitgevoerd te kunnen worden. Natuurlijk is de programmeur dit gewend en hij begint de “Appel” klasse verder uit te werken. In plaats van dat hij deze klasse abstraheert naar “Vrucht”, abstraheert hij deze klasse naar “Shape”. Hier begint de aard van de representatie opeens ook mee te spelen. De programmeur moet van zijn concept afstappen om de representatie technisch te kunnen representeren. Dit is hetzelfde principe als bij het schilderij van René Magritte “Ceci, n’est pas une pipe”. Het lijkt op een pijp, maar eigenlijk is het verf. Zo kan de programmeur wel het model van een appel overnemen, maar is altijd gebonden aan de aard van de computer en de software die erop draait. Wanneer een appel een twee dimensionale vorm op het beeldscherm wordt, krijgt hij  $x$  en  $y$  coördinaten toegewezen. Eigenschappen die hij eerst niet bezat. Dit is geen keuze, maar een noodzakelijkheid. Enerzijds is de programmeur zich bewust van hetgene dat een teken representeert, de betekenis van het teken, maar anderzijds is hij zich ook bewust van de vorm van het teken zelf. Met object georiënteerd programmeren wordt eigenlijk een concept uit het probleemdomen overgezet naar een nieuwe context welke met elkaar in de klasse gedeeltelijk samenvallen. De instanties krijgen een eigen identiteit die bestaan uit een eigen samenstelling. Wanneer deze instanties vanuit een Platoons perspectief bekeken worden is hun compositie wellicht veranderd en zijn het schimmen van andere Ideeën geworden.

## ***Samenvatting***

In dit hoofdstuk is uiteengezet hoe het Aristoteliaanse perspectief en het Platoonse perspectief gelijktijdig aan het werk zijn tijdens object georiënteerd programmeren. Alhoewel een Aristoteliaans model overgezet kan worden van het probleem domein naar een object model bestaande uit klassen, raakt daarbij de essentie verloren. Dit ligt voornamelijk aan de aard van het medium dat bepaalde eigenschappen van het object model vereist om te kunnen bestaan. Daardoor delen de instanties van het objecten in andere Ideeën dan de ‘realiteit’ waar ze uit zijn afgeleid.

## Conclusie

Dit onderzoek startte met de onderzoeksvraag: “Welke vooronderstellingen zijn er in object georiënteerd programmeren te onderscheiden wanneer het perspectief van Plato’s ideeënleer en Aristoteles logica wordt toegepast?” Object georiënteerd programmeren wordt vaak vergeleken met Plato’s ideeënleer, maar door kleine doch cruciale verschillen in de opvattingen daarover en de werkelijke ideeënleer bleek deze overeenkomst niet juist. Daarentegen was Aristoteles logica beter toepasbaar op het object georiënteerde concept. Hierbij staat het punt centraal dat er universele eigenschappen in de objecten zelf te vinden zijn die niet aan verandering onderhevig zijn. Door een de reductie van het abstraheren maakt deze logica echter ook iets anders van objecten dan dat ze zijn. Dit bleek ook het geval te zijn bij object georiënteerd programmeren. Het perspectief van Plato laat ons zien dat de objecten in de wereld altijd onderhevig zijn aan verandering en delen in universele Ideeën. Als we Plato’s ideeënwereld even buiten beschouwing laten als een plaats buiten ruimte en tijd, dan vertelt het ons dat sommige Ideeën niet gevonden kunnen worden in de dingen zelf. Door een Aristotelianse reductie worden we verder weggevoerd van de Idee, alhoewel we dan misschien wel in staat zijn om een universele structuur te onderscheiden. Wanneer we niet ons tevreden stellen met de veranderende dingen om ons heen en daarin delen van de Idee kunnen vinden is Plato’s ideeënleer zelfs in postmoderne tijden een waardevol perspectief. Programmeurs die object georiënteerd programmeren zouden hiermee ook een voordeel kunnen doen.

## Bronnen

Ray Giguette, "Building Objects Out of Plato: Applying Philosophy, Symbolism, and Analogy to Software Design." *Communications of the ACM*. 49 (2006): 66-71.

Alan Kay, "The Early History of Smalltalk." [1993].

<http://gagne.homedns.org/~tgagne/contrib/EarlyHistoryST.html> - 24-06-2008

Derek Rayside & Gerard T. Campbell, "An Aristotelian Understanding of Object-Oriented Programming." *OOPSLA '00*. Minneapolis: Minesota, 2000.

Allan Silverman, "Plato's Middel Period Metaphysics and Epistemology." [2003]. *Stanford Encyclopedia of Philosophy*. <http://plato.stanford.edu/entries/plato-metaphysics/> - 25-06-2008

### Internet discussies / blogposts:

Vlad Tarko, "The Metaphysics of Object Oriented Programming." [2006]. *Ubik Report*.

<http://ubikreport.blogspot.com/2006/12/metaphysics-of-object-oriented.html> - 24-06-2008

Cyocum, "Ancient Philosophy And Programming Languages." [2004]. *PerlMonks*.

[http://www.perlmonks.org/?node\\_id=349593](http://www.perlmonks.org/?node_id=349593) - 24-06-2008

Jason Salas, "Having trouble grasping object oriented programming? Read Plato." [2005].

*ASP.NET Weblogs*. <http://weblogs.asp.net/jasonsalas/archive/2005/03/28/396038.aspx> - 24-06-2008