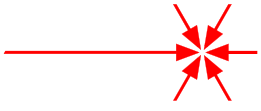
Deel 1. Universeel kernmodel

Universele grondslagen voor de modellering van systemen in het algemeen

Boekdeelwijzer

Onderwerp	*Hoofdstuk*
Wat zien we als we vanuit verschillende interesses naar een systeem kijken? Hoe kunnen we systemen weergeven? Kunnen we zelf bepalen wat we zien van een systeem?	Systeem en model
Is er een eigenschap die in systemen en modellen aanwezig is? Is het handig om dan zoveel mogelijk technieken op die eigenschap te baseren, zodat ze universeel toepasbaar worden?	Structuur
Wat is het nut van een bepaald systeem en hoe kan het systeem dat ter beschikking stellen? Bestaat er een eenduidige beschrijving van een systeem ook als het nog niet is gebouwd?	Functie en vorm
Hoe kunnen we zelf bepalen wat we wel en wat we niet van een systeem zien? Hoe worden modellen gemaakt? Wat voor soort modellen bestaan er en wanneer moeten ze worden toegepast?	Abstractie en typering



Boekdeelwijzer

Onderwerp	*Hoofdstuk*
Kunnen we de structuur van een systeem veranderen zonder dat de functie van het systeem verandert? Kunnen we het systeem elke gewenste structuur geven? Kunnen we een model op verschillende manieren noteren zonder dat het een ander systeem weergeeft?	Herstructurering
Zijn er duidelijke criteria op te stellen waarmee de kwaliteit van een systeem of een model kan worden beoordeeld? Wat is een goed systeem? Wat is een goed model? Hoe weten we of al onze herstructureringspogingen succesvol waren?	Kwaliteit van structuur

2. Systeem en model

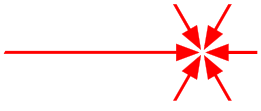
Hoofdstukwijzer

Onderwerp	*Paragraaf*
Wat is een systeem? Wat hoort wél en wat hoort níet tot een systeem?	Systeem, systeemgrenzen en functie
Waarom zien we nooit het systeem of de werkelijkheid in al zijn volledigheid? Wat zien we dan wel? Hoe verschilt een aspect van een deel? En van een gezichtspunt?	Gezichtspunten, aspecten en delen
Kunnen we een ‘op maat’ gemaakte weergave maken van een systeem zodat we er eenvoudiger mee om kunnen gaan? Is een model een aspect of deel van een systeem, of geen van beide?	Modellen
Hoe worden modellen dan gemaakt? Welke techniek is er voor nodig? Wat is het basis-principe achter het maken van een model?	Abstractie
Hoe verhouden de verschillende modellen en het systeem dat ze weergeven zich tot elkaar? Hoe staat de werkelijkheid naast een model van de werkelijkheid?	Weergavenruimte

2.1 Systeem, systeemgrenzen en functie

Wat zou de eerste definitie in de theorie van een ontwerpen van systemen moeten zijn? Dat moet natuurlijk de definitie van het lijdend voorwerp van alle moeite zijn. Dat is het systeem zélf. Een systeem is het eindpunt van een constructietraject. Een systeem is ook het beginpunt van een analysetraject. Het is dus zinvol om over een bruikbare definitie van het begrip systeem te beschikken. Van het begrip systeem zijn vele verschillende definities in omloop en de vraag is welke binnen de context van dit boek moet worden gekozen.

Wat is nu precies een systeem? Het definiëren van het begrip systeem is een soort ‘kip en het ei’-probleem. Om het ene begrip te definiëren zijn andere begrippen nodig en



omgekeerd. Voorlopig wordt daarom gekozen voor een definitie die in dit boek goed uitkomt en dat is dat een *systeem* een samenstel van een aantal dingen is, die nodig zijn om een bepaalde *functie* te kunnen bieden.

Een systeem is een groep dingen met een gezamenlijke functie.

De afhankelijkheid van die dingen van de functie van het systeem is belangrijk in deze definitie. Functie is de mogelijkheid die een systeem voor de gebruiker of observeerder heeft, of zou moeten hebben. Een kampeertent is bijvoorbeeld als een systeem te beschouwen. De functie die het systeem '*bestaansrecht*' geeft is 'het bieden van beschutting voor de kampeerders' en het zeildoek, de stokken, de haringen enzovoort vormen de 'dingen' van het systeem. Duidelijk is dat alle dingen nodig zijn om de kampeerfunctie te kunnen bieden. Een zaklantaarn bijvoorbeeld is geen ding van het systeem 'kampeertent' omdat die voor de functie 'bieden van beschutting' niet nodig is. Indien de kampeertent wordt gebruikt voor een andere functie, zoals het fungeren als windscherm, dan is een aantal dingen niet meer nodig en bestaat het nieuwe systeem enkel uit de dingen die noodzakelijk zijn voor deze functie.

Functie bepaalt wat wél en wat níet tot een systeem behoort.

Later zal de relatie tussen functie en dingen van het systeem meer in detail worden besproken. Tot dat moment is het belangrijk te beseffen dat wat wél en wat níet tot een systeem behoort, afhankelijk is van de functie van het systeem. De grenzen van het systeem worden dus bepaald door de functie van het systeem en niet zo zeer door fysieke of geometrische eigenschappen, hoewel later zal blijken dat deze natuurlijk ook zijn te herleiden tot de functie van het systeem.

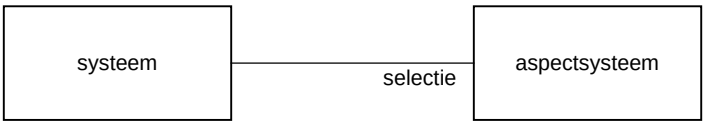
2.2 Gezichtspunten, aspecten en delen

Het is niet mogelijk om een voorwerp of een deel van de werkelijkheid in al zijn volledigheid (dus zoals het voorwerp of de werkelijkheid echt ís) waar te nemen. Daarom is het belangrijk te beseffen dat elke waarneming of afbeelding of gedachte in principe een incompleet beeld van het voorwerp betreft. Fysieke voorwerpen worden bijvoorbeeld altijd vanuit een bepaalde ruimtelijke richting bekeken. Afhankelijk van het ingenomen gezichtspunt kan dan een bepaald aanzicht worden gezien. Het gezichtspunt kan onbewust of bewust zijn gekozen maar altijd wordt er vanuit een bepaalde 'richting' gekeken en altijd is het beeld incompleet. De complete geometrie van een voorwerp kan worden bepaald door om het voorwerp heen te lopen en de verschillende aanzichten te combineren. Door in een bouwtekening verschillende aanzichten op te nemen, kan uiteindelijk de driedimensionale vorm van het voorwerp worden bepaald. Dus, hoewel het niet mogelijk is om alle kanten van het voorwerp tegelijkertijd te zien, is het wel mogelijk om door verschillende aanzichten te combineren een completer beeld te krijgen.

Hetzelfde gaat op voor de andere kenmerken van een systeem. Het niet mogelijk om alle kenmerken en eigenschappen van een systeem tegelijk te overzien. Van elk systeem kunnen verschillende aspecten worden beschouwd. Een *aspect* is een benoemd of afgebakend kenmerk. Door een systeem volgens een bepaald aspect, of een combinatie van aspecten, te analyseren wordt een incompleet beeld van dat systeem verkregen. Die incomplete waarneming heet een *aspectsysteem*. Een aspect is

een selectie van het systeem, zonder het systeem kunnen er geen aspectsystemen bestaan.

Een aspectsysteem is een selectie van het systeem volgens een bepaald aspect.



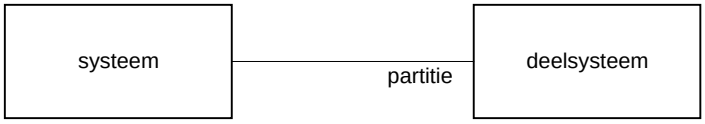
Figuur 1: een aspectsysteem is een selectie van een systeem

Verschillende, apart verkregen aspectsystemen moeten worden samengevoegd om een completer beeld van een systeem te verkrijgen, net als de verschillende aanzichten van een bouwtekening die samen de complete geometrie van het systeem beschrijven. Bijvoorbeeld bij een medisch onderzoek kunnen de beelden van een CT-scan worden aangevuld met de beelden van een MRI-scan om meer diagnostische informatie te verkrijgen.

Zowel voor de analyse als voor het ontwerp worden bepaalde aspecten gekozen. Welk aspect wordt gekozen, is afhankelijk van de interesse van de analist of ontwerper. De ontwerper van de logica van digitale chips beschouwt zijn chip als een verzameling Booleaanse formules, de elektronicus bekijkt zijn chip als een elektronische schakeling, terwijl de fysicus zijn chip vanuit de halfgeleidertechnologie beschouwt. Bij dit voorbeeld worden drie aspecten onderscheiden. Het is belangrijk om bij de analyse of ontwerp van een systeem het betreffende aspect expliciet te definiëren. Het is ook mogelijk om van een gekozen aspect weer deelaspecten te beschouwen. Bijvoorbeeld door eerst het elektrische schema te analyseren en vervolgens in te zoomen op het digitale gedeelte daarvan.

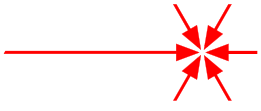
Behalve dat van een systeem slechts bepaalde aspecten kunnen worden beschouwd, kunnen ook bepaalde delen of partities apart worden genomen. Zo'n 'afgebakend geheel' van een systeem heet een *deelsysteem*.

Een complete partitie (met alle aspecten) van een systeem heet een deelsysteem.



Figuur 2: een deelsysteem is een partitie van een systeem

Een goede manier om aspectsystemen en deelsystemen uit elkaar te houden, is door de delen als puzzelstukjes en de aspecten als de kleuren te zien. Een puzzelstukje is een deelsysteem. Elk puzzelstukje is een deeltje van het totaal, het bevat in principe alle eigenschappen (alle kleuren) van het betreffende deel. Door de puzzelstukjes (in de juiste ordening) naast elkaar te leggen, ontstaat weer het complete systeem. Een monochromatische weergave van de puzzel is een aspectsysteem. Door de verschillende kleurlagen over elkaar te leggen ontstaat weer het complete beeld. Net zoals een kleurenfoto bestaat uit de aspectsystemen rood, groen en blauw.



Een deelsysteem bevat alle aspecten van een deel en een aspectsysteem bevat van alle delen hetzelfde aspect.

Een systeem is geen aspectsysteem en ook geen deelsysteem. Een systeem is het *supersysteem* van een aspectsysteem of deelsysteem. Wat voor de éne observeerder een systeem is, is echter voor een andere slechts een deelsysteem en voor weer een andere observeerder een aspectsysteem. Het onderscheid wordt bepaald door de context. Als er wordt gesproken van een aspectsysteem bestaat er in dezelfde context ook een systeem waar het betreffende aspectsysteem een selectie van is. En als er wordt gesproken van een deelsysteem bestaat er in dezelfde context ook een systeem waar het betreffende deelsysteem een partitie van is.

2.3 Modellen

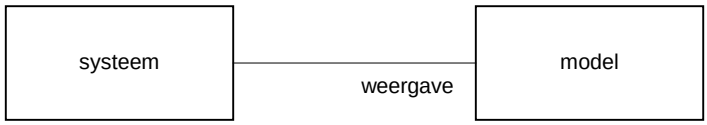
Van een werkelijk systeem kunnen niet alleen bepaalde aspecten worden geselecteerd, er kunnen ook speciale, ‘op maat gemaakte’ weergaven van worden gemaakt. Een dergelijke weergave heet een *model*. Modellen worden om verschillende redenen gemaakt. Nog niet bestaande systemen beginnen hun leven meestal als een model, bijvoorbeeld als bouwtekening of schets op een servetje of als schaalmodel. Modellen zijn ook een middel voor de ontwerper bij het stapsgewijs en controleerbaar ontwerpen van het uiteindelijke systeem. Van bestaande systemen kan met modellen worden bepaald wat het effect van gevaarlijke of schadelijke omstandigheden op het systeem zou zijn. Modellen zijn daarmee belangrijke hulpmiddelen voor zowel de ontwerper als de analist.

Vaak worden modellen gemaakt om het werk van de ontwerper of analist eenvoudiger te maken. De mens kan maar een bepaalde hoeveelheid informatie tegelijk overzien en met een model kan de hoeveelheid en het soort informatie worden gedoseerd. Een model is dus een vereenvoudigde weergave van een systeem.

Een model is geen *aspectsysteem*. Een model geeft wel een bepaald aspectsysteem of deelsysteem weer. Het verschil zit in het feit dat een model expliciet een nieuwe weergave is van het systeem en niet iets dat van het systeem apart is genomen. Een aspectsysteem is ‘gewoon’ een selectie van het werkelijke systeem, een model is een speciale weergave ervan. De elektrische bedrading van een huis vormt een aspectsysteem, een kamer van het huis vormt een deelsysteem, de bouwtekeningen vormen een model. Door van een auto onderdelen te verwijderen ontstaat een aspectsysteem of een deelsysteem, niet een model. Een model kan al bestaan voordat het systeem dat wordt weergegeven bestaat. Zonder systeem kan er echter geen aspectsysteem bestaan.

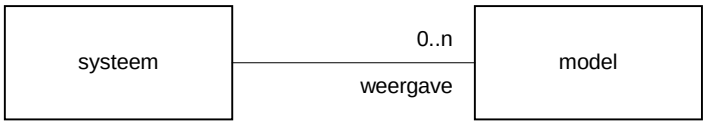
Een model bestaat dus beslist níet uit de delen of aspecten van het systeem. Als dat wel het geval zou zijn, dan zou het onmogelijk zijn een huis te ontwerpen zonder het huis zelf te bouwen…

Een model is een vereenvoudigde weergave van een systeem.



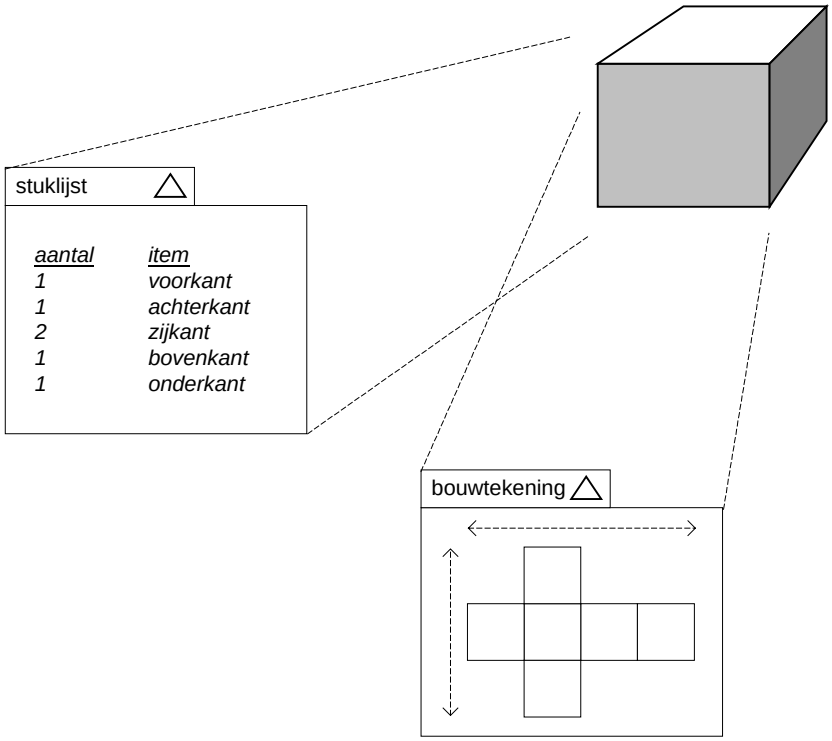
Figuur 3: een model is een weergave van een systeem

Van een bepaald systeem kunnen verschillende modellen worden gemaakt. Van een auto kan een bouwtekening worden gemaakt maar bijvoorbeeld ook een stuklijst, een rekenmodel maar ook een schaalmodel. De onderstaande figuur toont dat.



Figuur 4: meerdere modellen met hetzelfde systeem als onderwerp

Of, wat informeler weergegeven.



Figuur 5: een systeem kan door meerdere modellen worden weergegeven

2.3.1 Systeem en model

Een model is geen systeem maar net als tussen een systeem en een aspectsysteem is het onderscheid tussen model en systeem niet absoluut. Wat voor de één een model is,

is voor een ander misschien wel het systeem. Voor de ontwerper van een huis is een schaalmodel een model en het huis zijn systeem. Voor de fabrikant van schaalmodellen is het een systeem waarvan hij eerst modellen maakt. Een programmeur die de samenhang van zijn broncodebestanden onderzoekt, beschouwt die broncode op dat moment als een systeem, terwijl hij de broncode heeft geprogrammeerd onder de aanname dat het een model is van het softwaresysteem. Hij wisselt van gezichtspunt en daarmee krijgen sommige begrippen een andere inhoud.

Systeem en model zijn relatieve begrippen, afhankelijk van het ingenomen gezichtspunt en de gekozen context. Telkens als in dit boek over een systeem en het model daarvan wordt gesproken, bestaan er in dezelfde context een systeem als onderwerp van het betreffende model én een model als weergave van het betreffende systeem. Het model is niet het systeem, het onderwerp van het model is het systeem! Deze stelling is zeer consequent doorgevoerd in dit boek en zonder een goed begrip hiervan worden de belangrijke essenties van dit boek niet goed begrepen.

Een systeem is het onderwerp van een model.

Net als een systeem kan een model bestaan uit delen en aspecten. Een *deelmodel* is een integrale partitie van het model. Een *aspectmodel* is een selectie van het model op basis van een aspect.

2.3.2 Metamodel

Tussen het systeem en de modellen van dat systeem bestaat een strikte evenredigheid. Net als tussen die modellen onderling, hoewel die evenredigheid in praktijk niet expliciet zal worden beschreven. De opbouw van een model moet direct kunnen worden afgeleid uit eigenschappen van het systeem. De transformatie die daar voor nodig is, omvat bijvoorbeeld een nauwkeurig beschreven manier van abstraheren. Andersom zullen de elementen van een ontwerp of model aan de hand waarvan een systeem wordt gebouwd, moeten kunnen worden teruggevonden in het gebouwde systeem. De transformatie die nodig is om een model af te beelden op een systeem, moet de inverse zijn van de transformatie die is zijn om een systeem af te beelden op een model. De transformaties en elementen die nodig zijn om systeem en model van elkaar te kunnen afleiden, worden beschreven in een beschrijving die het *metamodel* wordt genoemd.

Een metamodel is de verzameling regels en elementen die de evenredigheid tussen een bepaald soort systeem en een bepaald soort model van dat systeem beschrijven.

2.4 Abstractie

Mensen kunnen de hoeveelheid informatie die ze te verwerken krijgen doseren door informatie selectief te beschouwen. Die techniek heet *abstractie*. Mensen creëren een mentaal model van de werkelijkheid door onbewust te abstraheren en kunnen, door de verschillende aspecten die zij kunnen waarnemen te combineren, een steeds completer beeld krijgen. Abstractie kan ook expliciet worden toegepast, namelijk bij het maken

van modellen. *Abstraheren* is het wegfilteren van details of kenmerken tijdens het opbouwen van een model. Door te abstraheren kan de ontwerper ervoor zorgen dat alleen de relevante informatie in het model wordt geplaatst.

> *Abstraheren is het opbouwen van een model en daarbij het zorgvuldig buiten het model houden van eigenschappen die niet relevant zijn voor het doel van het model. Modellen zijn altijd abstracties.*

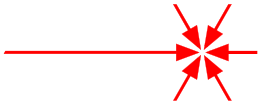
Abstractie kan op vele manieren worden toegepast. In de informatica gebeurt het meestal door werkelijke dingen te representeren met *informatische dingen* zoals records of objecten. Het nut van modellen is divers, maar belangrijk is dat een model niet fysiek hoeft te worden gebouwd, het hoeft alleen maar in 'informatische vorm' te bestaan. Een model maakt het simuleren van eigenschappen van een echt systeem mogelijk zonder het echte systeem eerst te moeten bouwen, of zonder het werkelijke systeem te moeten blootstellen aan potentieel schadelijke invloeden.

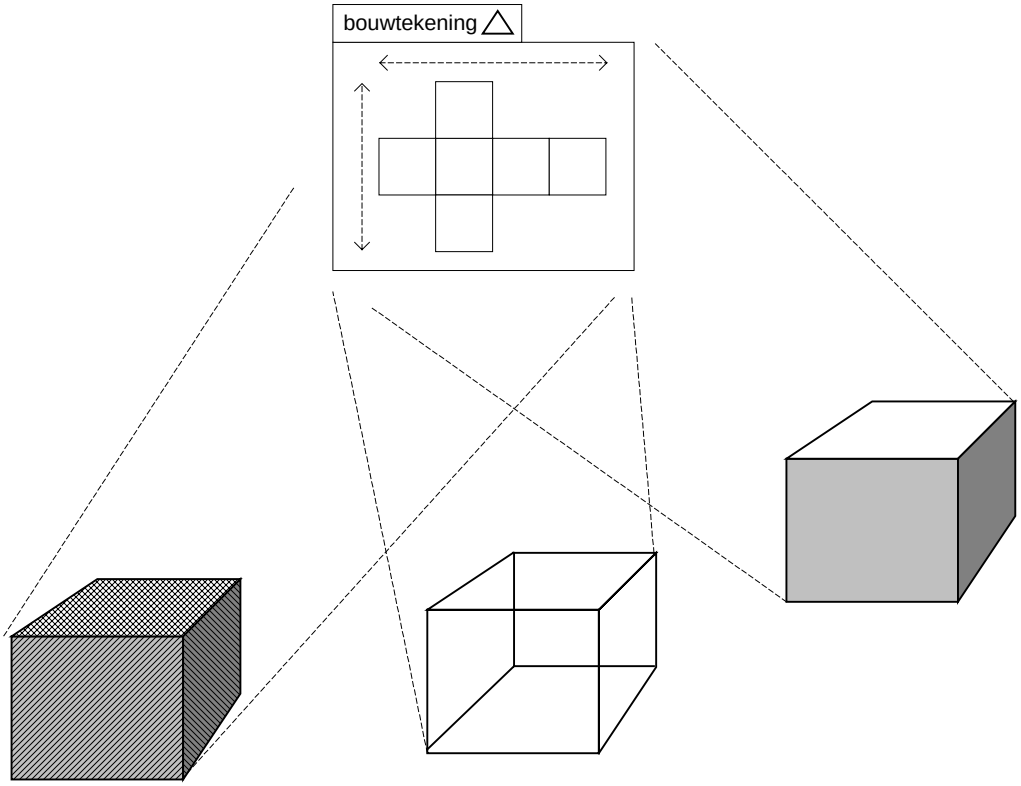
Of iets een systeem of een model is, wordt voornamelijk bepaald door het gezichtspunt dat wordt ingenomen. Vaak zijn softwaresystemen afspiegelingen van een (voorstelbare) werkelijkheid. Ten opzichte van die werkelijkheid is het software-systeem dan een model, maar voor de ontwerper van de softwaresystemen is het gewoon het systeem, het eindresultaat dat hij eerst moet modelleren. Het bouwen van dit soort softwaresystemen is te vergelijken met het namaken van een vliegtuig in Lego. Door de eenvoudige vorm van de Legoblokjes moet het vliegtuig noodzakelijkerwijs eenvoudiger zijn dan het werkelijke vliegtuig (abstractie). In plaats van Legoblokjes gebruikt de ontwerper van softwaresystemen echter virtuele bouwstenen, stukjes code, die informatiepatronen beschrijven. Op deze wijze kan een deel van de werkelijkheid worden nagemaakt in informatische vorm.

Abstraheren is het weglaten van informatie, niet het vervormen van informatie. Een abstractie is dus misschien niet compleet maar wel degelijk exact!

> *Abstracties zijn incomplete maar exacte weergaven van een systeem.*

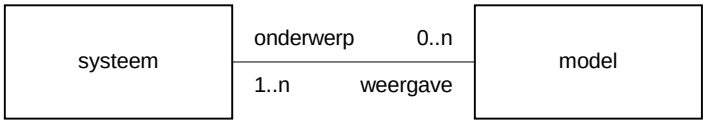
Een belangrijke constatering is dat een model altijd een abstractie is. Een model is altijd incompleet ten opzichte van het systeem. Een model dat natuurgetrouw alle details van het systeem weergeeft, ís het systeem. Juist omdat een model een incomplete weergave van een systeem is, worden niet alle aspecten of delen van een systeem weergegeven. Dat betekent dat een model geen onderscheid kan maken tussen twee systemen die juist op díe aspecten of delen van elkaar verschillen. Een bepaald model kan dus een weergave zijn van meerdere, verschillende systemen. De onderstaande figuur toont drie kubussen die op details verschillen maar door hetzelfde model kunnen worden weergegeven omdat die details in het model onbepaald zijn.





Figuur 6: een model kan meerdere systemen weergeven

In formele weergave ziet de verhouding tussen systeem en model er als volgt uit:



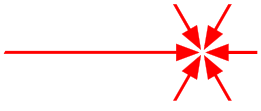
Figuur 7: een model is een weergave van een één of meerdere systemen, het systeem is het onderwerp van geen, één of meerdere modellen

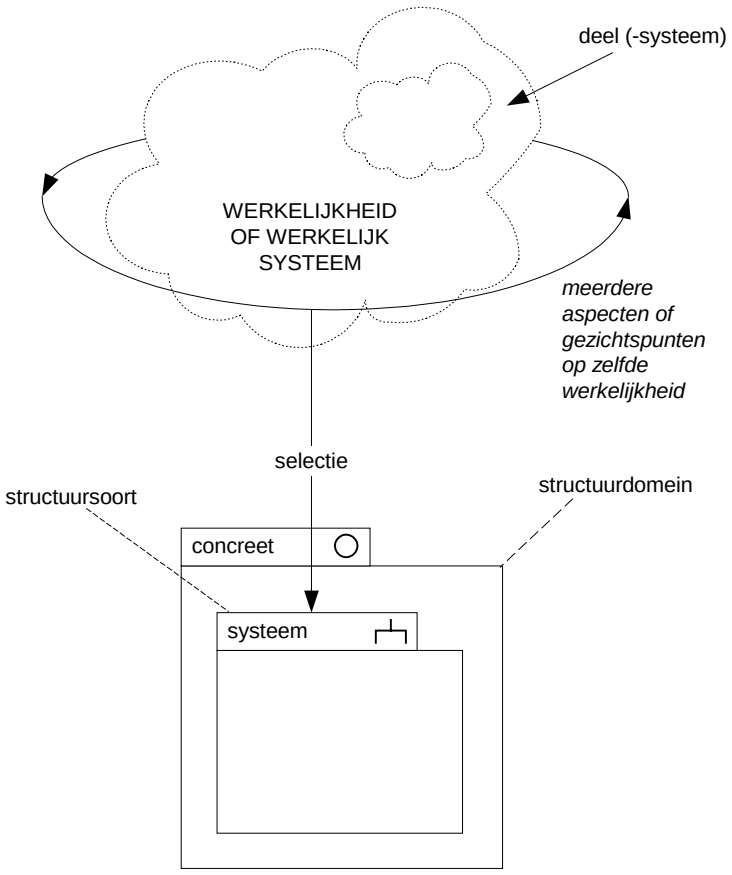
2.5 Structuurdomeinen

Modellen vormen een hulpmiddel voor de ontwerper om eenvoudiger met systemen te kunnen omgaan. Er zijn vele verschillende *weergaven* en *modellen* mogelijk van hetzelfde systeem, deze tonen dan allemaal hetzelfde systeem. En allemaal moeten ze daarom op één of andere manier verenigbaar zijn. Wijzigingen in de ene weergave van het systeem hebben soms consequenties voor de andere weergaven van het systeem. Als er modelelementen aan de bouwtekening worden toegevoegd, dan moet ook de stuklijst van het weergegeven systeem veranderen. Als er nieuwe dingen worden ontdekt aan een voorwerp, dan moeten de bijbehorende analysemodellen ook worden aangepast.

Soms heeft een verandering in één bepaalde weergave echter geen invloed op de andere weergaven. Of een ontwerper zijn bouwtekening wel of niet voorziet van middellijnen en symmetrielijnen, maakt voor de stuklijst niet uit. Om de verschillende modellen die de ontwerper eventueel maakt van hetzelfde systeem consistent te houden, moet de ontwerper weten hoe alle modellen samenhangen met elkaar en met het systeem. In dit boek zal daarvoor een figuur worden opgebouwd waarin uiteindelijk alle soorten modellen en technieken een plaats zullen krijgen. Naarmate het boek vordert, zal deze 'plattegrond van de weergavenruimte van het systeem' worden uitgebreid. De diverse technieken zullen daarbij hun specifieke plaats in het schema gaan innemen. Uiteindelijk zal de plattegrond drie weergavenruimtes of *structuurdomeinen* bevatten, gescheiden door de twee belangrijke *modellering-technieken*. De ontwerper kan in deze figuur zien hoe de verschillende technieken hem door de totale weergavenruimte van een systeem 'verplaatsen'.

Het eerste *structuurdomein* is de ruimte die 'de werkelijkheid' toont. Dit structuurdomein zal het *concrete domein* worden genoemd. Het concrete domein bevat de 'werkelijke' dingen (de systemen), alle andere structuurdomeinen bevatten beschrijvingen daarvan (de modellen). In principe kan de menselijke *perceptie* slechts aspectsystemen waarnemen, onze zintuigen zijn bijvoorbeeld ongevoelig voor het grootste deel van het spectrum van het elektromagnetische spectrum. Daarom kan worden beredeneerd dat het concrete domein dus 'slechts' aspectsystemen bevat. In dit boek is ervoor gekozen om níet die stelling in te nemen. Het concrete domein herbergt per definitie het systeem met al zijn aspecten, waarneembaar of niet. De onderstaande figuur toont hoe de werkelijkheid vanuit meerdere gezichtspunten kan worden gezien en hoe dat in principe leidt tot aspectsystemen in het concrete domein.

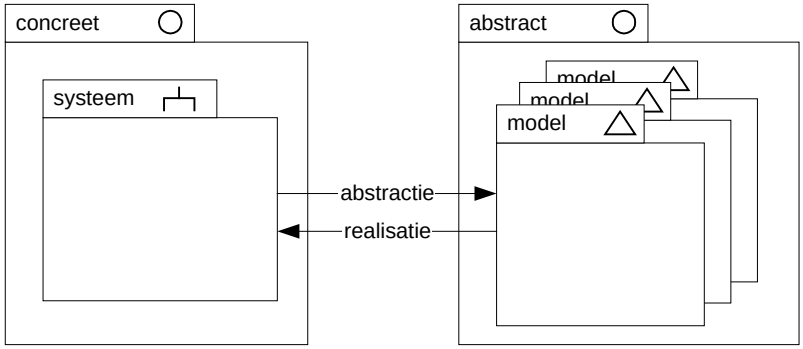




Figuur 8: in het concrete domein komt het systeem voor (selectie van de werkelijkheid)

Behalve systemen kunnen er ook modellen van een systeem bestaan. Door te abstraheren ontstaat bijvoorbeeld een abstraherend model. Naast een *concrete weergavenruimte* voor systemen (het concrete domein) bestaat er in de totale weergavenruimte van het systeem dus ook een *abstracte weergavenruimte* voor modellen (*het abstracte domein*). Deze twee ‘ruimtes’ zijn aan elkaar gerelateerd omdat het 'weergaven van hetzelfde systeem zijn en met de juiste technieken kan een ontwerper een handeling in het ene structuurdomein vertalen naar een handeling in het andere structuurdomein.

Een eenvoudig systeem kan direct in het concrete domein worden opgebouwd. Bij een complex systeem lukt dat niet, daarvan moeten eerst eenvoudige weergaven in het abstracte domein worden gemaakt. Deze abstracte weergaven of modellen kunnen dan worden gebruikt om het uiteindelijke systeem te *realiseren*. Realisatie is het daadwerkelijk bouwen van het systeem.



Figuur 9: modellen bestaan in een ander structuurdomein dan systemen

Verschillende handelingen zoals analyseren of ontwerpen doorlopen de weergaven in verschillende richtingen. Voor het bovenstaande figuur maakt dat niets uit, dat geeft samenhang aan, geen volgorde aan.

2.6 Samenvatting

Een selectie van een systeem op basis van een aspect of interesse heet een aspectsysteem. Een integraal deel van een systeem heet een deelsysteem. Deelsystemen kunnen worden gezien als ‘naast elkaar’ liggende dingen van het systeem. Een groep aspectsystemen kan worden gezien als ‘door elkaar’ of ‘over elkaar’ liggende ‘selecties’ van het systeem.

Een model is een vereenvoudigde weergave van een systeem en een systeem is daarmee het onderwerp van een model.

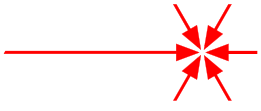
Er kunnen meerdere modellen bestaan die hetzelfde systeem weergeven. En omdat een model nooit alle delen en aspecten van een systeem weergeeft, kunnen er ook meerdere systemen bestaan die door hetzelfde model worden weergegeven.

Abstractie is een techniek die kan worden gebruikt om modellen te maken. Abstractie is het filteren van informatie en die in een aparte weergave plaatsen. Abstraherende modellen bevatten slechts de relevante informatie van een systeem.

Van een systeem kunnen vele verschillende weergaven worden gemaakt. De ontwerper of analist heeft de vrijheid om per handeling de meest geschikte weergave te kiezen. De verschillende weergaven zijn echter allemaal weergaven van hetzelfde systeem en moeten daarom consistent worden gehouden.

Weergaven van hetzelfde soort komen voor in hetzelfde structuurdomein. De ‘bruikbare’ dingen komen voor in het concrete domein, dit zijn systemen. De modellen of beschrijvingen daarvan komen voor in het abstracte domein, dit zijn modellen. In het concrete domein komen de ‘originelen’ voor, in het abstracte domein modellen daarvan.

Tussen systeem en model bestaat een verband dat wordt beschreven in het bij het model horende metamodel.



3. Structuur

Hoofdstukwijzer

Onderwerp	*Paragraaf*
We kunnen op verschillende manieren naar een systeem kijken, is er een eigenschap die in al die weergaven aanwezig is?	Structuur
Hoe ontstaat structuur, waar komt structuur vandaan? Hoe kunnen we de structuur van een geheel zichtbaar maken of ontdekken? Hoe vormen we delen tot een structuur of geheel?	Structureren
Wat maakt een structuur complex?	Structurele complexiteit
Hoe kunnen we structuur zodanig zichtbaar maken dat er geen structuren met een te hoge complexiteit zichtbaar worden?	Modulair en hiërarchisch structureren
Hoe wordt samenhang aangebracht tussen delen? Wat zorgt voor de samenhang?	Koppeling

3.1 Structuur

Om eenvoudig met de vele verschillende soorten systemen te kunnen omgaan, zonder voor elke soort systemen nieuwe technieken en principes te moeten ontwikkelen, moet er een soort algemene eigenschap worden gevonden. Een eigenschap die herkenbaar is in alle aspecten en delen van een systeem. Een eigenschap die bovendien ook nog eens in alle soorten systemen herkenbaar is. Als zo'n eigenschap bestaat, dan zijn de technieken die daarop worden gebaseerd universeel toepasbaar. Deze eigenschap is de *structuur* van een systeem. Structuur wordt gevormd door de delen (en aspecten) van een geheel en hun samenhang.

> *Structuur is de eigenschap van een geheel dat het uit delen (en aspecten) met onderlinge samenhang bestaat.*

Elk systeem of model ís in principe een structuur. Dezelfde 'aanwijsbare' structuur, zoals een speelgoedautootje of de tekening van een boom, kan zelfs voor de één een model zijn en tegelijkertijd voor de ander het systeem. Afhankelijk van welke

bedoelingen de betreffende persoon met de structuur heeft. Formeel kan worden gesteld dat door een structuur in het concrete domein te plaatsen de structuur als systeem wordt opgevat en door de structuur in het abstracte domein te plaatsen diezelfde structuur als model wordt opgevat. De betekenis van de handelingen die een ontwerper uitvoert op de structuur is gekoppeld aan het domein waarin hij de structuur plaatst.

Structuur kan in alle dimensies voor komen. *Ruimtelijke structuren* zijn natuurlijk de meest voor de hand liggende, maar het is ook mogelijk om over structuren van tijdgerelateerde dingen te spreken (temporele structuren). Bijvoorbeeld de samenhang van gebeurtenissen die plaatsvinden in een systeem. Of herhaling van gebeurtenissen en oorzaak-en-gevolg-relaties.

3.1.1 Principe van algemeenheid van structuur

Hoe er ook naar een systeem of model wordt gekeken, er is altijd structuur te ontdekken. Van dat feit kan op een bijzondere manier gebruik worden gemaakt. Op een structuur kunnen verschillende bewerkingen worden uitgevoerd en telkens zal het resultaat een nieuwe, kleinere of grotere structuur zijn. Dit is interessant en leidt tot een zéér belangrijk principe: op welke schaal we ook naar een systeem (of model) kijken en vanuit welk gezichtspunt we het systeem ook zien, telkens weer kan een structuur worden herkend en telkens weer kunnen dezelfde algemene handelingen of aanpassingen op die structuur worden verricht. Het is daarom mogelijk om algemeen geldige technieken te formuleren. Deze mogelijkheid wordt beschreven door *het principe van algemeenheid van structuur*:

> *Elk systeem of model is een structuur en door technieken zoveel mogelijk op de eigenschappen van structuur te baseren worden ze algemeen toepasbaar.*

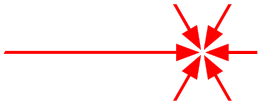
Dit principe is één van de meest ingrijpende principes die kan worden geformuleerd voor een systeem of model in het algemeen. Het principe maakt bovendien gebruik van een andere belangrijke eigenschap van structuur: *recursiviteit*. Op structuur kunnen handelingen worden verricht en omdat de uitkomst van die handelingen ook weer structuur moet hebben, kunnen op die uitkomsten weer dezelfde handelingen worden verricht.

> *De uitkomst van een bewerking op structuur heeft zelf ook weer structuur: structuur is 'recursief'.*

3.1.2 Het actieve en passieve aspect van structuur

Het is handig om in een structuur onderscheid te maken tussen twee aspecten: het *passieve* en het *actieve aspect*. Het actieve aspect betreft de dingen die data veranderen of uitwisselen. Deze zorgen voor de dynamische eigenschappen of verandering van de structuur. Het passieve aspect betreft de dingen die geen data veranderen of uitwisselen. Deze zorgen voor de passieve eigenschappen of toestand van de structuur.

> *Het actieve aspect van een structuur betreft de dingen die data uitwisselen of veranderen, het passieve aspect de dingen die dat niet doen.*



De fundering, muren en vloeren van een huis kunnen worden gezien als het passieve aspect van de structuur die we ‘huis’ noemen. De elektrische leidingen, de waterleiding en de verwarmingsbuizen kunnen worden gezien als het actieve aspect omdat ze materie of energie uitwisselen (elektriciteit, water, warmte).

De bepaling van het actieve en passieve aspect van een structuur is arbitrair en wordt gestuurd door het gekozen gezichtspunt. Zelfs in een schijnbaar passieve structuur als een hangbrug kunnen actieve aspecten worden benoemd. De delen die op elkaar leunen kunnen worden opgevat als actieve dingen omdat ze krachten uitwisselen. De ontwerper maakt díe keuze die hem het beste uitkomt.

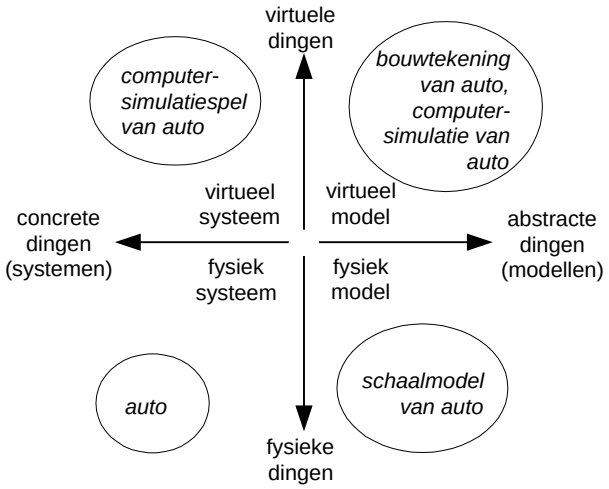
3.1.3 Structuurdomeinen

Er bestaan verschillende soorten structuur. Een systeem is een concrete structuur, een model is een abstracte structuur. Dezelfde structuur kan zelfs voor de één concreet zijn, maar voor de ander abstract. De bouwer van een miniatuurvliegtuig zal in zijn creatie een *concrete structuur* ofwel een systeem zien, en bouwtekeningen (modellen) maken om het te kunnen realiseren. De opdrachtgever die het miniatuurvliegtuig gebruikt om zich een beeld te kunnen vormen van een mogelijk nieuw product, zal het echter zien als model en dus als *abstracte structuur*.

Een systeem is een concrete structuur, een model is een abstracte structuur.

Concrete structuren zijn altijd ‘originelen’ of ‘eindresultaten’. Abstracte structuren zijn altijd modellen of weergaven van concrete dingen, abstracte structuren zijn daarom ook altijd incompleet.

Een structuur kan ook fysiek of virtueel zijn. Concreet wil namelijk niet altijd zeggen *fysiek*. Een systeem of model hoeft niet per se fysiek te zijn, denk maar aan rekenmodellen of computerprogramma's, die zijn niet fysiek, ze bestaan uit gegevens. Een rekenmodel is een *virtueel ding*. Fysieke dingen zijn *tastbaar*, virtuele dingen zijn niet tastbaar. Uiteindelijk moet alle *virtuele realiteit* een *fysieke drager* hebben, maar dat is niet wezenlijk. De onderstaande figuur toont de verschillende *verschijningsvormen van structuur*

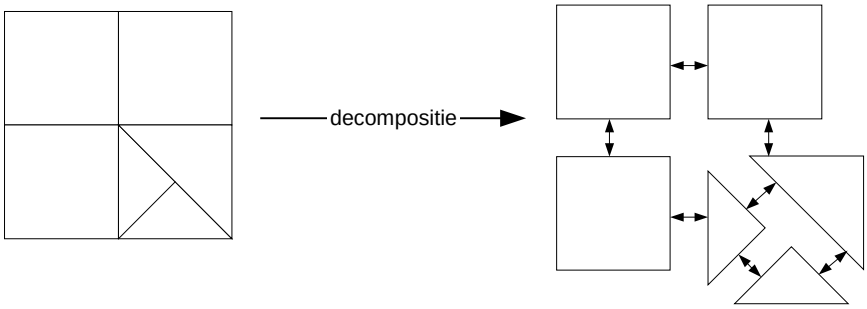


Figuur 10: structuur komt voor vier verschijningsvormen

3.2 Structureren

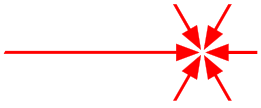
Om gebruik te kunnen maken van het uitgangspunt dat in elk systeem een bepaalde structuur is te herkennen, moeten er technieken worden geformuleerd die van toepassing zijn op structuur. Daartoe moet echter eerst die structuur worden bepaald. Dit kan op twee manieren. Allereerst door toepassing van *decompositie*. In een systeem kan een bepaalde opbouw worden gezien. Er zijn dan delen te vinden in een systeem die vanuit het gekozen gezichtspunt (schijnbaar) ondeelbaar zijn. Zulke delen geven het systeem een bepaalde structuur.

De structuur van een geheel kan worden ontdekt door toepassing van decompositie. Decompositie is een proces waarbij de delen van een geheel worden onderscheiden. De delen bestaan echter niet in isolatie, ze werken samen als een schijnbaar geheel dus elke keer als er delen worden onderscheiden, moeten ook de verbanden tussen die delen worden vastgelegd. Decompositie is daarom het afwisselend onderscheiden van delen en aanbrengen van koppelingen zodat uiteindelijk een structuur ontstaat.



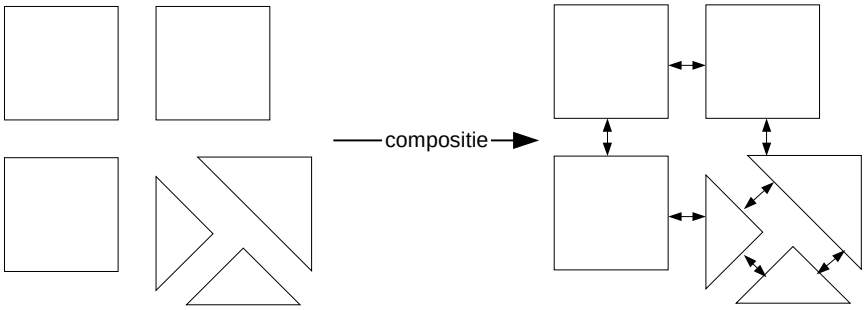
Figuur 11: decompositie is het onderscheiden van delen in een geheel

Decomponeren is het herkennen van delen en samenhang in een geheel.



Structuur kan natuurlijk aanwezig en gemakkelijk herkenbaar zijn, zoals een mens anatomisch gezien bestaat uit losse ‘delen’ zoals organen, botten en spieren. Structuur kan ook kunstmatig worden aangebracht of onzichtbaar zijn in het werkelijke systeem, zoals de grenzen van de landen van Europa. Het toepassen van decompositie maakt structuur.

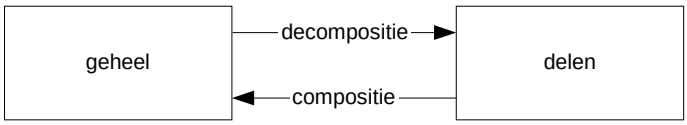
Behalve door toepassing van decompositie kan structuur ook worden bepaald door toepassing van *compositie*. Compositie is het koppelen van delen tot een nieuw geheel. Bestaande onderdelen worden samengevoegd tot een nieuw geheel. Vóór compositie waren er delen, ná compositie is er een geheel.



Figuur 12: compositie is het maken van structuur

> *Compositie is aanbrengen van samenhang tussen delen zodat ze een geheel gaan vormen.*

De onderstaande figuur toont de verhouding tussen compositie en decompositie.

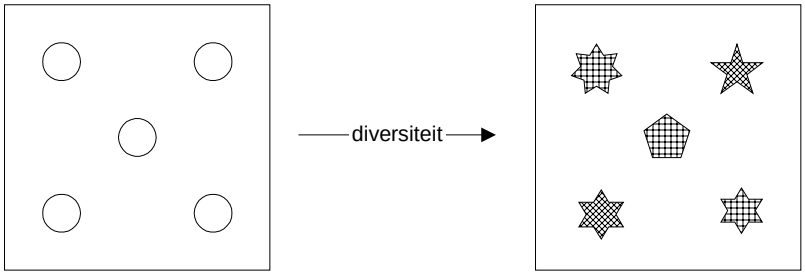


Figuur 13: decompositie en compositie zijn tegengestelde handelingen

> *Decompositie en compositie zijn structureringstechnieken: structuur ontstaat of wordt zichtbaar door decompositie of compositie.*

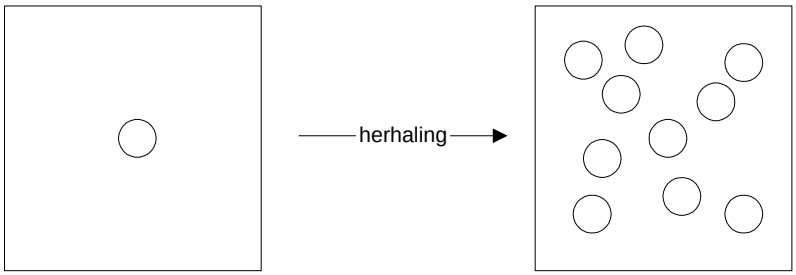
3.3 Complexiteit van structuur

Een eigenschap van structuur is dat naarmate deze groter en uitgebreider wordt deze ook moeilijker te begrijpen wordt. In elk systeem behalve het meest eenvoudige kunnen delen van diverse opmaak en werking worden ontdekt. Hoe meer verschillende delen, hoe lastiger het is om het overzicht te bewaren. Er ontstaat namelijk door die diversiteit steeds meer informatie en er moet met steeds meer verschillen rekening worden gehouden. Diversiteit vergroot de keuze die we hebben bij het maken van inschattingen en beslissingen. Hoeveel eenvoudiger zou een spelletje Snookerbiljart niet zijn als alle ballen dezelfde kleur hadden? *Diversiteit* maakt een structuur dus complexer.



Figuur 14: diversiteit maakt een structuur complexer

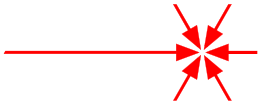
Behalve verschillende delen bestaan er ook nog gelijksoortige delen. Gewoon meer van hetzelfde wat wordt aangegeven door de factor herhaling. Een structuur die bestaat uit tien dezelfde delen is eenvoudiger dan een structuur die bestaat uit duizend dezelfde delen. Alle handelingen die we op zo'n grote structuur kunnen verrichten duren langer en zijn moeilijker naarmate de hoeveelheid handelingen groter is. Naast diversiteit maakt ook *herhaling* een structuur complexer.



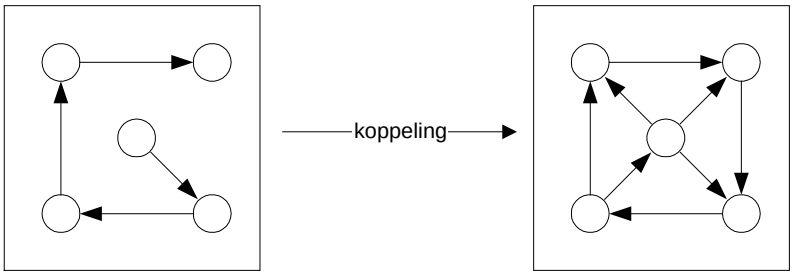
Figuur 15: herhaling maakt een structuur complexer

De delen waaruit een structuur is opgebouwd, werken samen aan een gezamenlijk gedrag. Om dit te kunnen bereiken, zijn ze onderling verbonden of gekoppeld. Als de delen onderling zijn gekoppeld, is het moeilijk om het gedrag van een enkel deel in isolatie te besturen. Elk deel gebruikt voor zijn werking andere delen die op hun beurt ook weer andere delen gebruiken. Hoe meer *koppelingen* er aanwezig zijn in een structuur, hoe meer delen er moeten worden beschouwd bij het bepalen van de uitwerking van een bepaalde lokale gebeurtenis. En dus hoe groter de betrokken hoeveelheid informatie.

Een bekend spelletje kan dit illustreren. Mikado is een spel waarbij een grote hoeveelheid dunne, lange stokjes willekeurig wordt opgestapeld, bijvoorbeeld door een bundel van die stokjes gecontroleerd uitéén te laten vallen. De kunst is om stokjes uit de stapel te verwijderen zonder dat er andere stokjes bewegen. Elke deelnemer neemt om beurten een stokje uit de stapel. Het gemakkelijkst te verwijderen zijn de stokjes zijn die het minste contact hebben met andere stokjes. De moeilijkste stokjes zijn de stokjes die het meeste contact hebben met andere stokjes. Omdat die andere stokjes zelf ook weer contact hebben met andere stokjes is het lastig om te zien hoe een beweging van een bepaald stokje de andere stokjes zal laten bewegen. Een kleine beweging van een individueel stokje kan een kettingreactie van bewegingen door de hele stapel veroorzaken. Om een stokje succesvol uit de stapel te verwijderen, moeten



we weten welke stokjes afhankelijk zijn van het betreffende stokje. *Koppeling* maakt, net als diversiteit en herhaling, een structuur complexer.

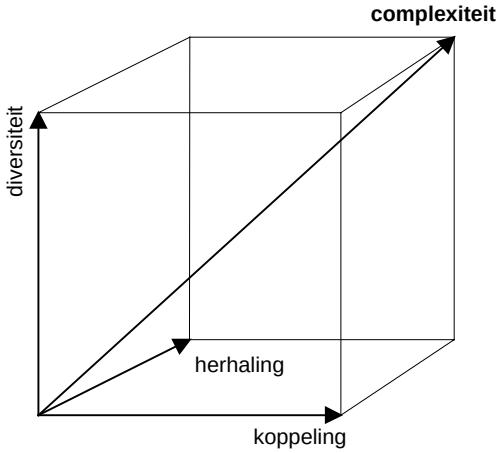


Figuur 16: koppeling maakt een structuur complexer

De *complexiteit* van een structuur zou dus wel eens bepaald kunnen worden door de factoren diversiteit, koppeling en herhaling. Voor de bouw van een ruimteschip van Lego bijvoorbeeld is een groot aantal bouwstenen nodig die voor een groot deel verschillend zijn (diversiteit) en waarbij de bouwsteentjes in een bepaalde samenhang (koppeling) moeten worden gebracht. Hoe groter de factoren diversiteit, koppeling en herhaling, hoe ingewikkelder de opbouw van het ruimteschip zal zijn. We kunnen dus zeggen dat het ruimteschip, afhankelijk van de grootte van de factoren, een bepaalde complexiteit heeft. Kenmerkend voor het samenspel van de afzonderlijke factoren is dat naarmate ze groter zijn er meer gegevens moeten worden beschouwd om het voorwerp te begrijpen. De complexiteit van een structuur lijkt daarmee te worden bepaald door de totale *hoeveelheid informatie* die nodig is om het systeem of een deel ervan te kunnen bestuderen.

Complexiteit van structuur wordt veroorzaakt door herhaling van delen, de diversiteit van delen, en koppeling van delen.

De onderstaande figuur toont schematisch hoe herhaling, diversiteit en afhankelijk samen voor een bepaalde *structurele complexiteit* zorgen:



Figuur 17: structurele complexiteit is de resultante van de koppeling, diversiteit en herhaling

Belangrijk bij de toepassing van deze definitie is het besef dat het een ‘rekentruc’ is en niet per se de waarheid. Complexiteit is misschien wel te complex om ooit in model te kunnen worden gebracht. Het is een verzonnen definitie die past in de rest van het systeem en die daadwerkelijk kan helpen om uiteindelijk betere softwaresystemen te maken.

3.4 Modulair en hiërarchisch structureren

Door het toepassen van decompositie wordt de structuur van het systeem zichtbaar. In principe wil de ontwerper door het uitvoeren van een decompositie het systeem op een hoger niveau van detaillering weergeven. Bijvoorbeeld omdat de verschijnselen waarin hij geïnteresseerd is op dat bepaalde niveau van detaillering het beste zichtbaar zijn. Het is echter niet altijd mogelijk om direct, in één keer, een geheel tot een structuur met de gewenste mate van detail te decomponeren. Een wolkenkrabber decomponeren door meteen alle boutjes, steentjes, plankjes en stopcontacten in kaart te gaan brengen is onmogelijk. Of op z’n minst ambitieus. Als bijverschijnsel van het toepassen van decompositie wordt namelijk ook complexiteit zichtbaar. Er blijken ineens verschillende en gelijkende soorten delen te zijn: diversiteit en herhaling. Er blijkt ineens ook samenhang tussen die delen te zijn: koppeling.

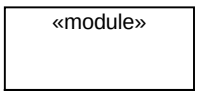
Structureren (decomponeren, componeren) vergroot de complexiteit van een structuur.

Decompositie, en ook compositie, maakt dus structurele complexiteit zichtbaar en dat kan lastig zijn bij verdere bewerkingen op de structuur. Om tijdens de decompositie niet teveel complexiteit in één keer zichtbaar te maken, moet de decompositie telkens in zodanig kleine stapjes worden toegepast dat het resultaat telkens net niet te complex wordt. Dus zodanig dat er per stap niet teveel structuur wordt ontdekt. Het hulpmiddel hierbij is *hiërarchische decompositie*. Door stapsgewijs de delen die het resultaat waren van een ‘normale’ of *modulaire decompositie* zelf ook weer te decomponeren, ontstaan delen met verschillende niveaus van *aggregatie*. Tijdens een hiërarchische decompositie kan de ontwerper de complexiteit van het ene *niveau van aggregatie* naar het andere als het ware gecontroleerd opvoeren.

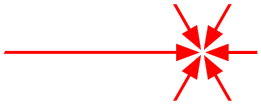
Decompositie en compositie (structurering) kunnen worden uitgevoerd op verschillende niveaus van aggregatie.

Hiërarchisch structureren maakt dat delen altijd weer gehelen kunnen vormen en gehelen altijd weer uit delen kunnen bestaan. Om het verschil tussen deel en geheel onbepaald te kunnen laten, wordt het woord *module* gebruikt.

Een module is een deel óf een geheel (onbepaald). Structuur bestaat uit modules.



Figuur 18: een module



Het hoogste aggregatieniveau is altijd de structuur als geheel. Op dit niveau wordt er nog geen opbouw in het systeem onderkend. Op een volgend niveau van aggregatie kunnen de meest algemene modules worden onderscheiden. Vervolgens kan *recursief* elke module op zich ook weer worden gedecomponeerd. Op het laagste niveau van aggregatie zijn de atomen van het systeem zichtbaar, de kleinste modules. De structuur wordt niet tijdens elke recursie concreter of abstracter, hij wordt slechts steeds meer of minder gedetailleerd (geaggregeerd) weergegeven.

Het verschil tussen modulaire en hiërarchische decompositie en compositie is (bewust) wat kunstmatig aangebracht want in de praktijk zijn de twee verschillende 'soorten' niet los van elkaar toe te passen. Er zijn echter wel duidelijk twee verschillende aspecten in een *structureringsproces* te herkennen en het verschil tussen die twee aspecten is essentieel. Deze twee aspecten zijn:

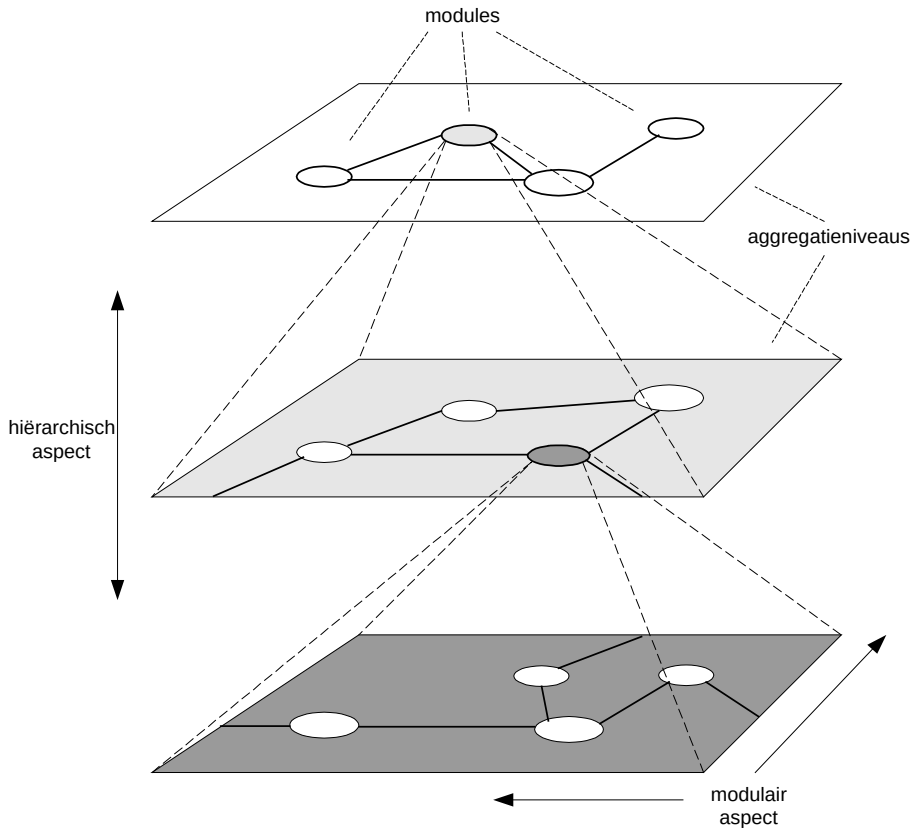
- Het hiërarchische aspect

Bij een *hiërarchische* of *recursieve decompositie* (of compositie) worden delen afzonderlijke weer beschouwd als gehelen en gedecomponeerd tot aggregaten van 'deel-delen'. Door het toepassen van hiërarchische decompositie ontstaan verschillende *niveaus van aggregatie* én de daarmee samenhangende *structurele hiërarchie*.

> *Een structuur die bestaat uit verschillende niveaus van aggregatie heeft structurele hiërarchie, of kortweg hiërarchie.*

- Het modulaire aspect

Bij een *modulaire* of *platte decompositie* (of compositie) wordt de onderlinge samenhang van delen van een gegeven niveau van aggregatie bepaald of ontdekt.



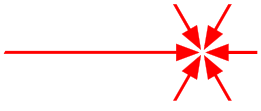
Figuur 19: structurering heeft een modulair en een hiërarchisch aspect

> *Modulair betekent: gestructureerd op een enkel niveau van aggregatie.*
> *Hiërarchisch betekent: gestructureerd over meerdere niveaus van aggregatie.*

3.4.1 Het mononiveau

Hiërarchische decompositie kan niet oneindig worden doorgezet. Op een gegeven moment is het laagste niveau bereikt, het elementaire niveau. Op dit niveau van aggregatie zijn de delen enkelvoudig en kunnen deze niet verder worden gedecomponeerd. Het laagste niveau van aggregatie heet het *mononiveau*.

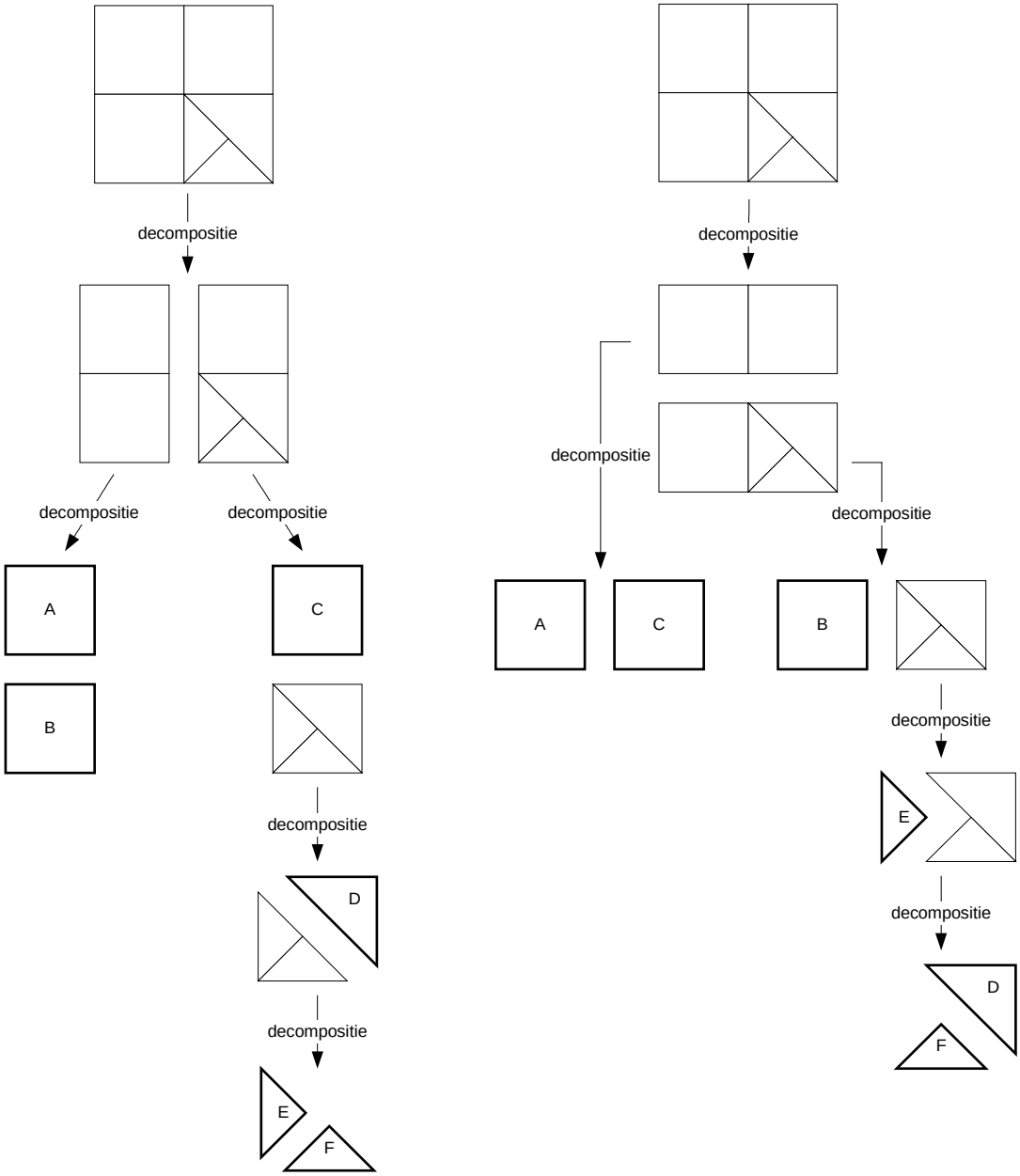
Om niet te belanden in onhandige en slechts theoretisch interessante situaties, waarbij decomposities tot op het ‘subatomaire’ niveau werden doorgevoerd, moet worden gerealiseerd dat in de meeste systemen een bepaald niveau van aggregatie bestaat dat als mononiveau gekózen kan worden. Voor het bouwen van huizen is een baksteen nog interessant, maar waar de baksteen zelf van is gebouwd is niet meer interessant. Voor het analyseren van de menselijke anatomie is het orgaanniveau nog interessant, waar de organen zelf van zijn gemaakt vaak niet meer. Het mononiveau is dus het laagste niveau van aggregatie dat vanuit het gekozen gezichtspunt nog interessant is.



Voor een programmeur van softwaresystemen is dat vaak het broncodeniveau van zijn hogere programmeertaal zoals BASIC. Dat de BASIC-instructies zelf weer zijn opgemaakt uit instructies op machinecodeniveau die op hun beurt weer uit instructie op microcodeniveau bestaan is niet relevant. Voor een constructeur is het mononiveau vaak het niveau van zijn beschikbare bouwstenen.

Bij de analyse van een bestaand systeem valt op dat de structuur van het mononiveau onafhankelijk is van het aantal tussenniveaus van aggregatie dat nodig was om het mononiveau te bereiken. De structuur van het mononiveau is ook onafhankelijk van de structuur van elk van die tussenniveaus. Dit betekent dat als je twee monteurs een auto uit elkaar laat halen, beiden zouden eindigen met dezelfde bouwstenen terwijl de volgorde die beiden gebruiken om de auto te demonteren misschien wel compleet verschillend is.

De onderstaande figuur laat twee verschillende decomposities van hetzelfde geheel zien die uiteindelijk op het laagste niveau van aggregatie dezelfde delen blootleggen.



Figuur 20: de structuur van het mononiveau is onafhankelijk van de wijze van decompositie

Deze eigenschap van structuur kan worden gebruikt om, uitgaande van een gegeven mononiveau (de bouwstenen), verschillende tussenniveaus te bedenken of maken zonder dat er een ander eindresultaat verschijnt. Belangrijk is om te beseffen dat de tussenniveaus hulpmiddelen zijn, die gebruikt kunnen worden om mensen niet te overweldigen met complexiteit.

3.4.2 Hoeveel niveaus van aggregatie?

Bij het decomponeren van een structuur ligt de structuur van het laagste niveau van aggregatie dus vast, dat is het mononiveau waarop alle bouwstenen individueel zichtbaar zijn. De structuur van het hoogste niveau ligt ook vast, dat is het systeem als geheel waar geen enkele bouwsteen individueel zichtbaar is. De structuur van alle tussenliggende niveaus is arbitrair, die kan, binnen grenzen, naar wens worden gekozen.

> *Bij een compositie of decompositie is de structuur van elk tussenniveau van aggregatie binnen grenzen arbitrair.*

Een decompositie kent minimaal twee niveaus van aggregatie: het systeemniveau en het mononiveau of *bouwsteenniveau*. Een decompositie kan in maximaal zoveel aggregatieniveaus worden uitgevoerd als er elementaire modules zijn (*mono's*) minus één. In dat geval wordt bij elke recursie slechts één enkel deel extra onderscheiden. Het aantal niveaus waarin een decompositie daadwerkelijk wordt uitgevoerd, kan vrij worden gekozen en is afhankelijk van de complexiteit van de diverse structuren die ontstaan of zichtbaar worden. In principe worden er zoveel niveaus van aggregatie gekozen als er nodig zijn om complexe structuren te vermijden.

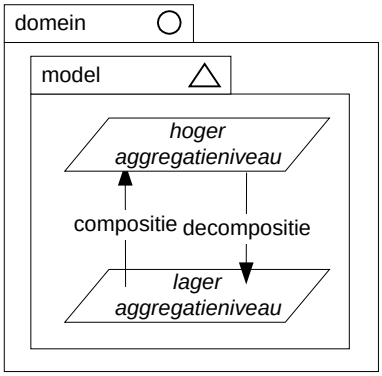
> *Een structuur moet in zoveel niveaus van aggregatie worden gestructureerd (gedecomponeerd of gecomponeerd) als nodig zijn om complexiteit te vermijden.*

Een automonteur die wordt gevraagd om een auto zodanig te ontmantelen tot losse onderdelen dat de auto later ook weer in elkaar kan worden gezet, zal niet in één keer alle individuele onderdelen loshalen en op de grond uitstallen. Intuïtief zal hij eerst de motor, de carrosserie, de versnellingsbak en de andere grote delen van elkaar scheiden. Vervolgens zal hij dan deze deel voor deel uit elkaar halen. Stapsgewijs zal hij de auto op verschillende niveaus van aggregatie decomponeren. Het aantal niveaus dat hij daarbij gebruikt, is zodanig gekozen dat hij voor zijn gevoel niet wordt overweldigd door de aantallen en koppelingen tussen de onderdelen.

Van tevoren kan worden geschat hoeveel niveaus van aggregatie er ongeveer nodig zullen zijn om de structuur van een voorwerp overzichtelijk te kunnen tonen. Als het voorwerp bijvoorbeeld uit circa vijftienhonderd onderdelen bestaat - dus het mononiveau vijftienhonderd modules telt - en er wordt gesteld dat elke deelstructuur maximaal twintig delen mag tonen (om de complexiteit te beheersen) dan zijn er vier niveaus nodig. Het eerst niveau toont het geheel. Het volgende niveau, het tweede, toont twintig delen. Al die delen kunnen zelf op een volgend niveau ook weer uit twintig delen bestaan, zodat er op het derde niveau maximaal vierhonderd delen zichtbaar kunnen zijn. En op vierde niveau, tenslotte, is ruimte om alle vijftienhonderd delen te tonen. Dat is dan het mononiveau. Nu is twintig een arbitraire keuze maar in de praktijk zullen het er ongeveer zoveel zijn. Misschien wat minder. Er wordt wel eens gesteld dat een mens maximaal zeven dingen tegelijkertijd zou kunnen relateren. In dat geval zouden er vijf niveaus nodig zijn. (1 (=7^0), 7 (=7^1), 49 (=7^2), 343 (=7^3) < 1500 < 2401 (=7^4) .)

3.4.3 Structurering en structuursoorten

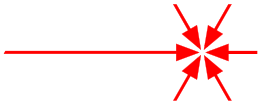
Decompositie en compositie zijn gebonden aan de gekozen structuursoort. Door te decomponeren of te componeren ontstaat niet ineens een ander soort systeem of model; het resultaat van beide technieken is van dezelfde structuursoort als het uitgangspunt. De ontwerper blijft binnen de gekozen soort structuur dus de decompositie van een gebouw resulteert niet ineens in een stuklijst of een schaalmodel, slechts in een onderdelen van een gebouw. En de decompositie van een fiets resulteert ook niet ineens in een model van die fiets, slechts in de onderdelen van die fiets. Decompositie en compositie zijn in alle verschillende weergaven van het systeem mogelijk en de ontwerper kan díe weergave kiezen die hem het beste uitkomt om de decompositie of compositie in te verrichten maar moet er rekening mee houden dat de verschillende weergaven onderling wel consistent moeten blijven.

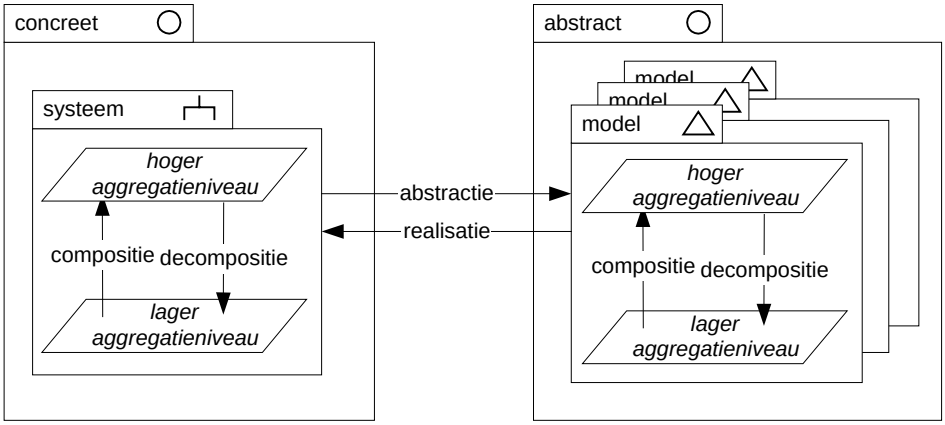


Figuur 21: decompositie en compositie zijn gebonden aan een specifieke structuursoort

> *Decompositie en compositie zijn technieken die een structuur op verschillende niveaus van aggregatie kunnen tonen zonder daarbij van aspect of structuursoort te veranderen.*

De onderstaande figuur toont hoe de ontwerper met decompositie en compositie kan ‘navigeren’. De figuur toont ook dat decompositie en compositie niet kunnen worden gebruikt om van een bepaald soort model een ander soort model te maken, daarvoor zijn andere technieken nodig (zoals abstractie).





Figuur 22: decompositie en compositie zijn technieken voor gebruik binnen een bepaalde structuursoort en een bepaald structuurdomein

In een bepaald model kunnen natuurlijk meerdere aspecten van een systeem zijn gecombineerd, maar bij de decompositie of compositie van die structuur zal altijd één aspect sturend zijn. Dat aspect heet het *dominante aspect*.

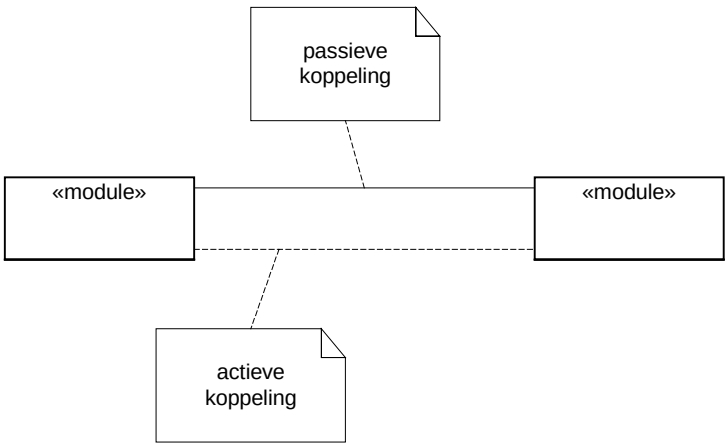
3.5 Koppeling

Structuur bestaat uit modules (delen en of gehelen) en koppelingen. Modules kunnen op verschillende manieren gekoppeld zijn.

3.5.1 Actieve en passieve koppeling

Koppeling kan actief of passief zijn. Als modules door de koppeling materie, gegevens of energie uitwisselen, is er sprake van een *actieve koppeling*. Als modules gekoppeld zijn maar door die koppeling géén materie, gegevens of energie uitwisselen, is er sprake van *passieve koppeling*.

Een actieve koppeling zal worden getekend als een onderbroken lijn, een passieve koppeling zal worden getekend als een doorgetrokken lijn.



Figuur 23: een passieve en een actieve koppeling

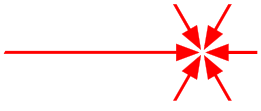
De koppeling tussen de postcode van een woonplaats, straat en huisnummer is een passieve koppeling: er worden geen data uitgewisseld. De koppeling tussen een automotor en de benzinetank is actief. De koppeling tussen een 100m-sprinter en het startschot ook, er wordt informatie (het startsignaal) uitgewisseld.

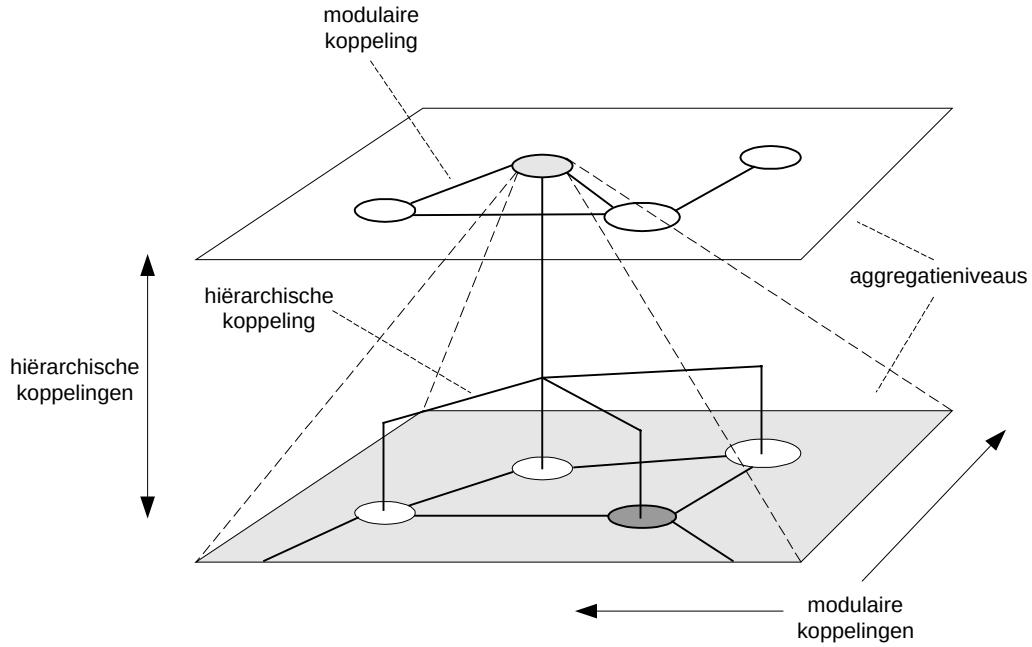
> *Een koppeling waarbij materie, gegevens of energie wordt uitgewisseld, is een actieve koppeling. Een koppeling waarbij géén materie, gegevens of energie wordt uitgewisseld, is een passieve koppeling.*

Een actieve koppeling kan worden onderscheiden van een passieve koppeling door te onderzoeken of er informatie, materie of energie wordt uitgewisseld met de koppeling of niet.

3.5.2 Modulaire en hiërarchische koppelingen

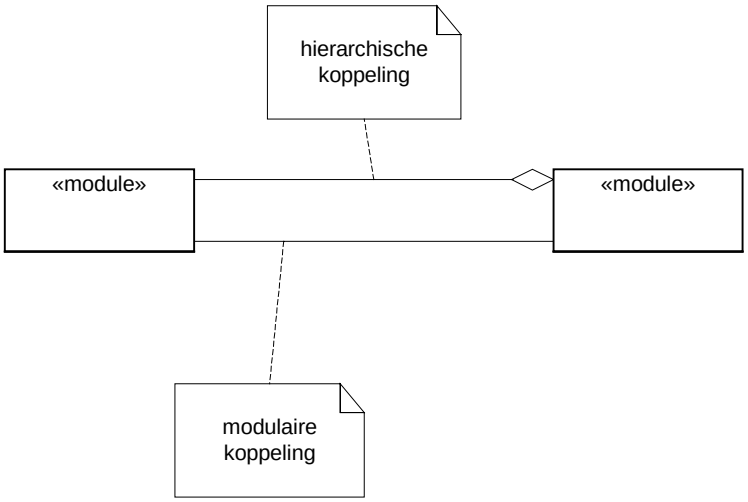
Delen hebben een ander soort koppeling met elkaar dan met het geheel waartoe zij behoren. Het geheel ‘bestaat’ op een ander niveau van aggregatie dan de delen dus de *koppelingen* tussen delen en geheel lopen van het éne niveau naar het andere. De onderstaande figuur laat dit informeel zien:





Figuur 24: koppelingen kunnen modulair of hiërarchisch zijn

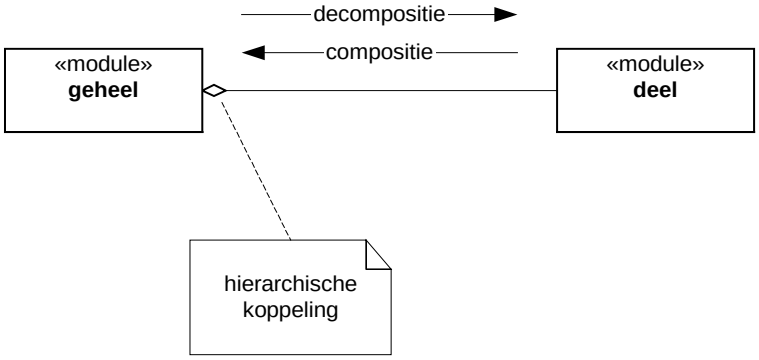
Een koppeling die modules van verschillende niveaus van aggregatie met elkaar verbindt, heet een *hiërarchische koppeling*. Een *koppeling* die modules op hetzelfde niveau van aggregatie met elkaar verbindt, heet een *modulaire koppeling*. Naar analogie van hiërarchische en modulaire structurering.



Figuur 25: een modulaire en een hiërarchische koppeling

Koppelingen tussen modules van verschillende niveaus van aggregatie (delen en geheel) heten hiërarchische koppelingen. Koppelingen tussen modules van hetzelfde niveau van aggregatie heten modulaire koppelingen.

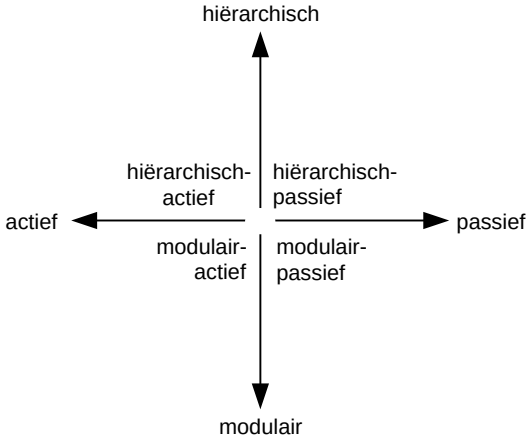
Een modulaire koppeling kan worden onderscheiden van een hiërarchische koppeling door te onderzoeken of het éne deel kan worden gevonden door het andere deel te decomponeren of componeren (wat in dat geval dus 'een geheel' blijkt te zijn). Als dat het geval is, is er sprake van een hiërarchische koppeling, anders van een modulaire koppeling. De koppeling tussen een fietser en zijn fiets is modulair: een fiets is geen deel van de fietser en een fietser is ook geen deel van een fiets.



Figuur 26: een hiërarchische koppeling is te onderscheiden van een modulaire door te onderzoeken of de éne module een (de-) compositie van het andere is

3.5.3 Overzicht van koppelingen

De onderstaande figuur toont de verschillende soorten koppelingen in samenhang.



Figuur 27: koppelingen kunnen worden verdeeld in vier soorten

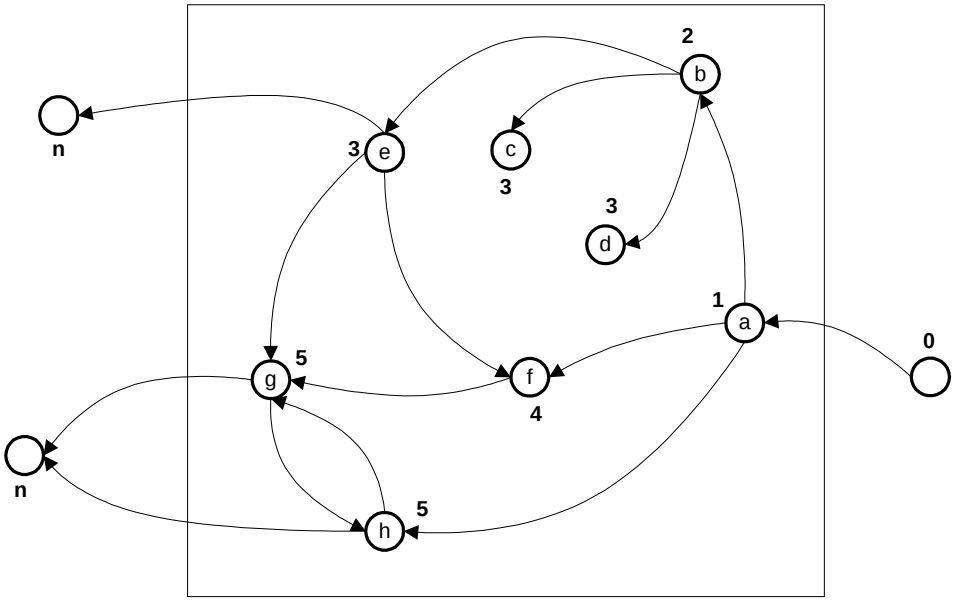
3.6 Volgorde in structuur

Bij het wijzigen, bekijken of maken van modules (files, classes, functies enzovoort) die afhankelijk zijn van elkaar is er altijd een optimale volgorde om dat te doen. In elke afhankelijkheidsstructuur kan namelijk een ‘natuurlijke’ volgorde worden onderscheiden. Deze volgorde kan worden getoond door het expliciet aanbrengen van *volgnummering*. Het aanbrengen van volgnummering is een belangrijke bewerking op een structuur die later in verschillende contexten zal terugkomen. De volgnummers zijn bijvoorbeeld een belangrijk gegeven bij het kiezen van de volgorde waarin de delen van een nieuw systeem moet worden gebouwd. De volgnummering geeft de volgorde en richting aan waarin effecten zich door het systeem voortplanten en volgorde waarin het systeem moet worden opgebouwd, getest en veranderd.

Elke structuur heeft een natuurlijke volgorde waarin effecten zich door de structuur voortplanten.

De regels die moeten worden gevolgd om een structuur van volgnummers te voorzien zijn eenvoudig:

- Modules die niet tot de betreffende structuur horen maar wel direct daardoor worden gebruikt, de aanbieders van het systeem, hebben *volgnummer 0*.
- Modules die niet tot de betreffende structuur horen maar die wel direct gebruiken, de afnemers van het systeem, hebben per definitie *volgnummer n* (letter n).
- Modules die wel tot de betreffende structuur horen maar geen interne afhankelijkheden hebben, hebben *volgnummer 1*.
- Modules die tot de betreffende structuur horen en ook nog eens een interne afhankelijkheid hebben krijgen als volgnummer *één nummer hoger dan de modules met het hoogste volgnummer waarvan ze zelf afhankelijk zijn*. Het volgnummer van een dergelijke module is daarmee ook altijd de langste afstand gemeten in 'aantal' afhankelijkheden naar laag 0.
- Indien de structuur cyclische afhankelijkheden bevat, *dan krijgen alle modules in die daardoor gekoppeld zijn hetzelfde volgnummer*.
- Indien de structuur ongerichte koppelingen bevat, *dan krijgen alle modules in die daardoor gekoppeld zijn hetzelfde volgnummer*.



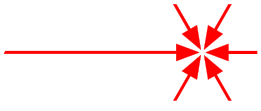
Figuur 28: de volgnummering van een gerichte graaf

Enkele consequenties van het op deze manier aanbrengen van volgorde zijn:

- Alle modules met hetzelfde volgnummer vormen een *laag.*
- Een module is nooit afhankelijk van modules met een hoger volgnummer.
- Directe en indirect *cyclische afhankelijkheden* (lussen) maken de betreffende structuur minder volgordelijk en meer monolithisch.
- De volgnummers zijn géén identificerende nummers, er kunnen dubbele volgnummers voor komen.

Op basis van volgnummers kan ook worden voorspeld wat voor gevolgen wijzigingen in de structuur hebben:

- Een module met een bepaald volgnummer afhankelijk maken van een module met lager volgnummer geeft nooit lussen en verandert de volgnummers niet. Deze wijziging beïnvloedt de bestaande structuur dus niet.
- Een module met een bepaald volgnummer afhankelijk maken van een module met hetzelfde volgnummer geeft nooit lussen maar verandert de volgnummering.
- Een module met een bepaald volgnummer afhankelijk maken van een module met een hoger volgnummer geeft lussen en verandert de volgnummering. Dit is een ingrijpende wijziging voor de bestaande structuur.



3.7 Samenvatting

Structuur is de organisatie van de delen van een geheel. Structuur is een eigenschap die in vrijwel alle systemen en weergaven van systemen is terug te vinden en daarom geschikt om te dienen als basis voor algemene technieken en principes.

Eén van de belangrijke grondslagen daarbij is het *principe van algemeenheid van structuur*. Dit principe stelt dat elk systeem een structuur is en dat de uitkomst van een bewerking op een structuur zelf ook weer een structuur is.

Compositie is het 'mechanistisch' in elkaar zetten van een systeem. Decompositie is het 'mechanistisch' uit elkaar halen van een systeem. Beide technieken doen structuur ontstaan.

Structureringstechnieken	
Compositie	Compositie aanbrengen van koppeling tussen delen zodat ze een geheel vormen.
Decompositie	Decomponeren is het herkennen (benoemen) van structuur in een geheel.

De structuur die door compositie of decompositie zichtbaar wordt, heeft ook een bepaalde complexiteit. Structurele complexiteit kan worden gezien als het resultaat van drie factoren:

Factoren van structurele complexiteit	
Koppeling	Koppeling is het aantal verbindingen dat bestaat tussen de delen van een structuur.
Diversiteit	Diversiteit is het aantal ongelijksoortige delen in een structuur.
Herhaling	Herhaling is het aantal gelijksoortige delen in een structuur.

Om de hoeveelheid complexiteit niet in één compositie- of decompositieslag zó groot te maken dat elke verdere analyse of beschouwing hinder zou hebben van die complexiteit, is het verstandig compositie of decompositie hiërarchisch uit te voeren. Door hiërarchisch te structureren kan op gecontroleerde wijze het systeem in steeds kleinere deelsystemen worden opgedeeld, zodanig dat elk volgend aggregatieniveau met eenvoudig resultaat aan een modulaire decompositie kan worden onderworpen. Gebruik hiërarchische structurering telkens wanneer een enkele modulaire structureringsstap een te complexe structuur zichtbaar zou maken of zou opleveren. Maak dus onderscheid tussen twee aspecten van structurering:

Structureringsaspecten

Structureringsaspecten	
Modulair	Met modulaire structurering wordt bedoeld: de structurering van een delen op een enkel niveau van aggregatie, zonder daarbij delen als geheel te beschouwen.
Hiërarchisch	Met hiërarchische structurering wordt bedoeld: het overgaan naar een lager of hoger niveau van aggregatie door de delen van een bepaald niveau weer als geheel te decomponeren of andersom.

Decompositie en compositie zijn technieken die gebonden zijn aan een bepaalde weergave van een systeem. Door decompositie of compositie toe te passen wordt het systeem niet vanuit een ander gezichtspunt zichtbaar, alleen op een ander niveau van aggregatie. Decompositie en compositie vergroten de structurele complexiteit.

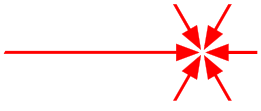
Om het verschil tussen geheel en deel onbepaald te laten (een deel is meestal zelf weer een geheel en een geheel bestaat meestal zelf weer uit delen), wordt het woord *module* gebruikt.

Een structuur bestaat uit modules en koppelingen.

Er kunnen verschillende soorten koppeling worden benoemt.

Koppelingsoorten	
Modulair	Een modulaire koppeling verbindt delen van een hetzelfde aggregatieniveau.
Hiërarchisch	Een hiërarchische koppeling verbindt delen van een verschillend aggregatieniveau. Feitelijk verbind een hiërarchische koppeling delen met gehelen.
Actief	Een actieve koppeling wordt gebruikt voor de uitwisseling van materie, gegevens of energie.
Passief	Een passieve koppeling wordt níet gebruikt voor de uitwisseling van materie, gegevens of energie.

Afhankelijkheid is een actieve koppeling tussen aanbieder en afnemer.



4. Functie en vorm

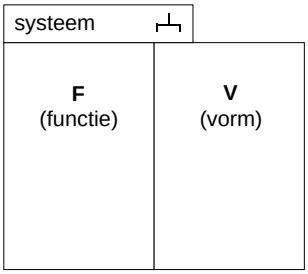
Hoofdstukwijzer

Onderwerp	*Paragraaf*
Waarvoor bestaat een systeem, wat is zijn nut en hoe kan het systeem dat ter beschikking stellen?	Functie en vorm
Waarom bestaat een systeem uit de dingen waaruit het bestaat? Welk criterium bepaalt de structuur van een systeem, ook van een nog niet bestaand systeem?	Bestaansafhankelijkheid en cohesie
Welke soort structuur is in alle systemen te herkennen?	Principe van algemeenheid van structuur
Als er behoefte is aan een bepaalde functie en die functie kan nauwkeurig worden beschreven, waarom moet er dan toch nog een heel ontwikkelingstraject volgen?	Principe van onbepaaldheid van vorm
Waarom is de uitkomst van zo'n ontwikkeling-traject niet uitsluitend of uniek, waarom zijn er altijd meerdere oplossingen mogelijk?	Principe van meervoudigheid van vorm
Wat zijn de uitgangspunten van de constructie van een systeem? Waar moeten we, behalve over de gewenste functie, over beschikken voordat het daadwerkelijke ontwikkelingstraject kan beginnen?	Palet
Wat is een ontwerptraject en wat is de principiële opbouw daarvan? Als we dan de functie en het palet kennen, hoe moeten we dan de juiste vorm bepalen? Wat is een functiemodel en waarvoor dient het?	Ontwerptraject

4.1 Functie en vorm

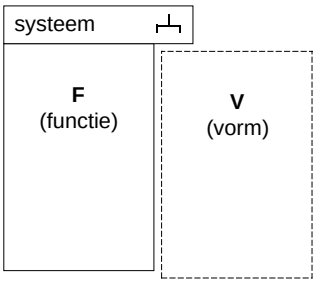
Aan de éne kant bestaat een systeem uit (meestal) tastbare onderdelen. Dit zijn de onderdelen waaruit het systeem is opgebouwd: de moeren en de bouten. Of de broncode als het gaat om een computerprogramma. Deze soort opbouw van een systeem is zijn *vorm*. Aan de andere kant heeft een systeem ook een bepaalde toepassing of bruikbaarheid. De bruikbaarheid van een systeem is zijn *functie* of *taak*. Vorm maakt het mogelijk om van functie gebruik te kunnen maken. Vorm is als het ware een middel en functie het doel.

> *Elk systeem heeft een bepaalde functie. Functie is het gedrag dat een systeem in staat is te vertonen. Elk systeem heeft naast functie ook een bepaalde vorm. Vorm is de drager van een functie: vorm maakt het mogelijk om van functie gebruik te kunnen maken.*



Figuur 29: een systeem heeft functie én vorm

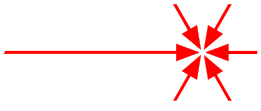
Een systeem dat nog moet worden gebouwd, heeft nog geen vorm maar wél een (toekomstige) functie. Functie is daarmee een belangrijk gegeven bij het ontwerpen van nieuwe systemen.



Figuur 30: een systeem dat nog moet worden gebouwd heeft nog geen vorm

Vorm is niet het uiterlijk of de esthetische vormgeving van een systeem, met vorm wordt in dit boek expliciet ‘de drager van functie’ bedoeld.

Vorm en functie zijn *concreet*: het zijn eigenschappen van het werkelijke systeem. Concreet wil niet altijd zeggen *fysiek*. Vorm hoeft namelijk niet per se fysiek te zijn en functie is nooit fysiek. Een computerprogramma in uitvoering is niet fysiek of tastbaar, het is *virtueel* (maar toch concreet). Toch is het computerprogramma een vorm met een bepaalde functie.



Functie en vorm zijn concreet maar niet per se fysiek.

4.1.1 Functie, subroutine en procedure

Het woord ‘functie’ is wat verwarrend, het heeft in de praktijk geen eenduidige betekenis. Programmeurs van computerprogramma’s zien een functie vaak als een stuk code dat ze intikken, kunnen ‘aanroepen’ vanuit andere functies en parameters kunnen meegeven om een bepaald resultaat te kunnen verkrijgen. In de wiskunde wordt een functie weergegeven als R=Functie(x,y,z) en duidt het een geparameteriseerde berekening aan. Daarnaast komt functie voor als:

- Zijn functie in dat bedrijf is...
- De sluis heeft als functie het water...
- Deze apparatuur heeft als functie...

Het woord functie heeft uit zichzelf dus geen eenduidige betekenis maar kenmerkend bij alle voorbeelden is dat er een ding is dat een bepaald gedrag kan vertonen respectievelijk moet kunnen vertonen. De functie Sinus is in staat om op basis van een parameter een resultaat te geven. Een digitale thermometer is in staat invoer, in de vorm van de warmte van het medium waar de sensor thermisch contact mee heeft, om te zetten in uitvoer in de vorm van een gemeten waarde in Kelvin, Celsius of Fahrenheit. Zijn functie of functionaliteit is het weergeven van temperatuur.

Het is de mogelijkheid om een bepaald gedrag te vertonen dat in dit boek wordt aanduidt als functie. In dit boek is een functie dus per se niet een stuk code, maar meer een stuk functionaliteit van een systeem. Een functionele decompositie is dus niet een decompositie in subroutines of procedures maar een decompositie op basis van functionaliteit of mogelijkheden.

4.2 Bestaansafhankelijkheid en cohesie

Waarom heeft het motorblok van de gemiddelde auto geen achteruitkijkspiegels? Of een handige bekerhouder voor warme en koude drankjes? Waarom hoort een krukas wel tot het motorblok en de motorkap niet? Het antwoord op deze vragen is dat het betreffende systeem, de motor, een bepaalde *functie* heeft en in die specifieke samenstelling het best kan voldoen aan die functie. De functie heeft voor de betreffende fysieke opbouw gezorgd. Zonder de functie van het geheel hebben de deelvormen namelijk geen bestaansrecht. De functie van een motorblok is het leveren van bewegingsenergie. Een achteruitkijkspiegel of een motorkap is daarbij niet nodig. Zonder deze onderdelen kan de motor nog steeds functioneren. Andersom echter hebben de onderdelen die wél tot de motor behoren zonder de functie waaraan zij meewerken geen bestaansrecht.

Een systeem heeft een bepaalde functie en alle vorm of onderdelen die het systeem nodig heeft om die functie te kunnen leveren, hebben een bepaalde relatie tot die functie. Deze relatie wordt *bestaansafhankelijkheid* genoemd.

Elke functie of vorm die zijn bestaansrecht ontleent aan een andere functie of vorm is daarvan bestaansafhankelijk.

Met het begrip bestaansafhankelijkheid kunnen goede en slechte ‘gehelen’ worden beoordeeld. Een geheel is van hoge kwaliteit als alle delen bestaansafhankelijk zijn van de functie van het geheel. Van een dergelijk geheel wordt gezegd dat hij een hoge *cohesie* heeft.

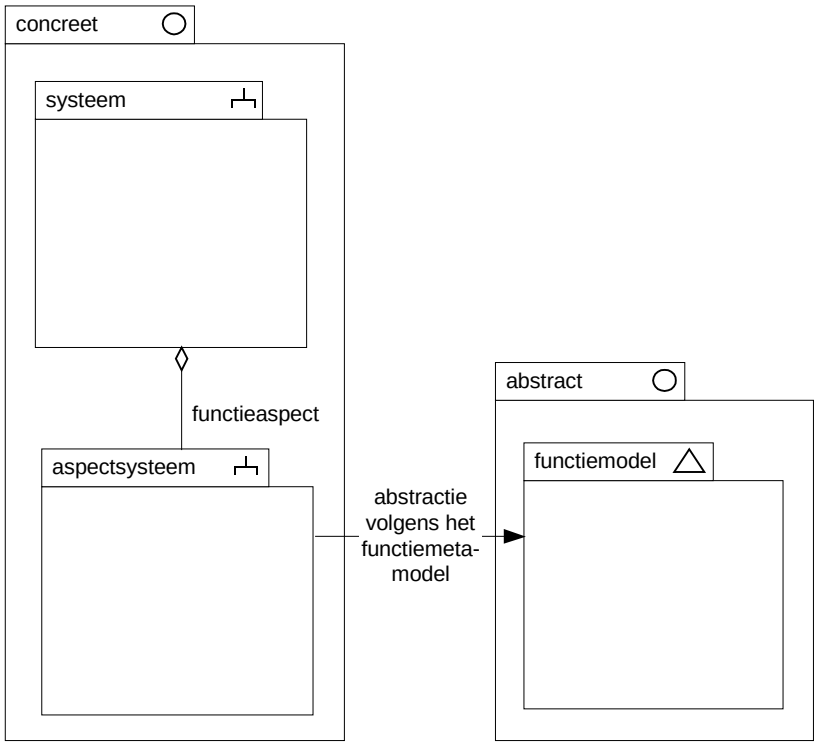
Een geheel is coherent als alle delen bestaansafhankelijk zijn van de functie van het geheel.

4.3 Principe van algemeenheid van functie

Van elk systeem (dit boek beschouwd alleen dingen met enige vorm van bruikbaarheid als systeem) kan een structuur van functies worden bepaald. Dit maakt functie de bindende factor tussen systemen van verschillende vorm en vormsoort.

Elk systeem, ongeacht hoe het gebouwd is en welke vorm het heeft, heeft een functieaspect. Dit is het principe van algemeenheid van functie.

Deze structuur van functies heet het *functieaspectsysteem* van het systeem. Een ontwerp van dit functieaspectsysteem heet een *functiemodel*. Het is dit model waar de ontwerper aanvankelijk naar op zoek is. De onderstaande figuur toont de samenhang.



Figuur 31: een functiemodel is een abstractie van het functieaspect van een systeem

4.4 Principe van onbepaaldheid van vorm

Een systeem heeft een bepaalde functie. In elke functie kan weer een structuur van functies worden ontdekt. Door elke functie recursief te decomponeren in deelfuncties is het mogelijk het systeem stapsgewijs in steeds meer detail te specificeren. In principe kan dit proces van functionele verfijning tot op het 'atomaire' niveau worden voortgezet. Het resultaat van elke nieuwe stap is echter steeds hetzelfde: een nieuwe decompositie of compositie van functies. Op elk niveau is de functie belangrijk, niet de vorm. Steeds het ‘wat’, niet het ‘hoe’.

De functionele structuur van een systeem is een beschrijving die steeds verder kan worden verfijnd maar het blijft functie en op geen enkel niveau van aggregatie wordt het vorm. Decompositie of compositie veranderen alleen maar het niveau van aggregatie van de gekozen weergave, niet het soort weergave of het weergegeven aspect. Functionele decompositie leidt dus ook niet uiteindelijk naar de gewenste vorm van het systeem. Aan de functie van een systeem kan niet worden gezien welke vorm of opbouw een systeem heeft of zou moeten hebben.

De functie van een systeem doet geen uitspraak over de vorm van het systeem. Dit is het principe van onbepaaldheid van vorm.

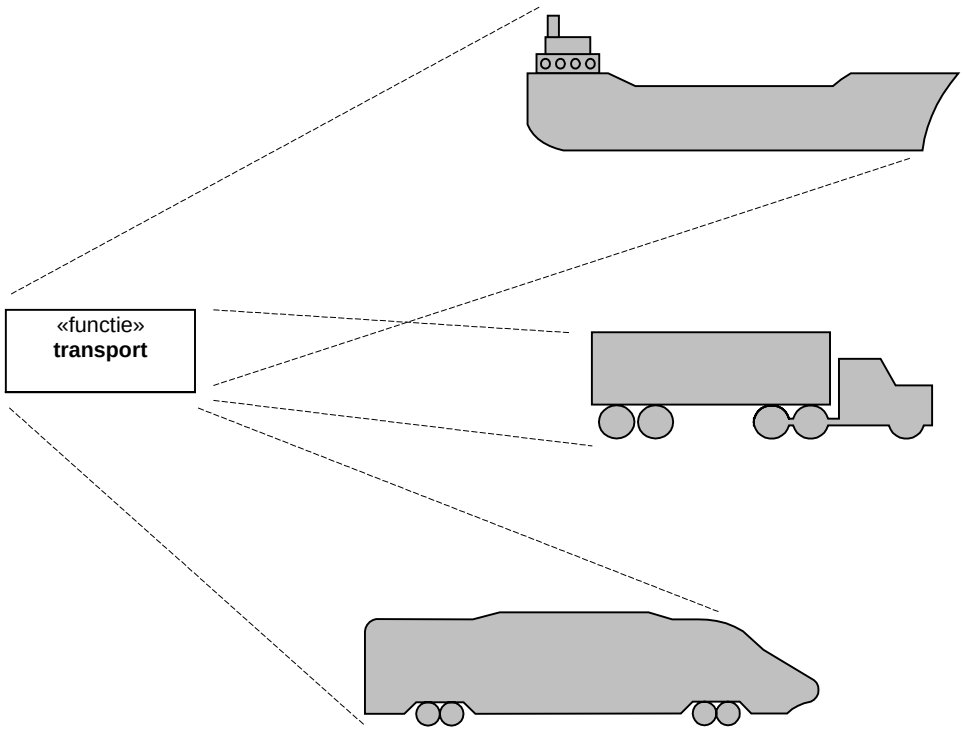
4.5 Principe van meervoudigheid van vorm

De concrete onderdeeltjes van het systeem vormen in hun actuele samenstelling slechts één van de vele denkbare vormen die kunnen voldoen aan de functie van het systeem. De functie van een systeem staat los van de actuele verschijningsvorm van het systeem. Een passief signaalfilter wordt gebouwd met weerstanden, condensatoren en spoelen. Een actief signaalfilter wordt gebouwd met transistoren. Een digitaal signaalfilter zelfs met IC’s en software. Toch kunnen alle filters zijn gebouwd voor dezelfde functie. Er zijn altijd meerdere vormen denkbaar die dezelfde functie kunnen hebben. De meeste vormen zijn natuurlijk ook te misbruiken voor meerdere functies maar die situatie zal niet worden beschouwd.

Figuur 32: dezelfde functie kan altijd op meerdere manieren worden vormgegeven

Als er meerdere vormen mogelijk zijn met dezelfde functie, dan is het ook mogelijk om een vorm te veranderen zonder dat de functie verandert. Deze bijzondere relatie tussen functie en vorm wordt beschreven door *het principe van meervoudigheid van vorm*.

Een bepaalde functie kan door meerdere vormen worden geleverd en een vorm kan worden veranderd zonder dat de functie ervan verandert. Dit is het principe van meervoudigheid van vorm.



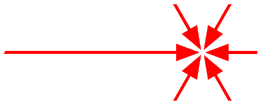
Figuur 33: verschillende vormen met dezelfde functie

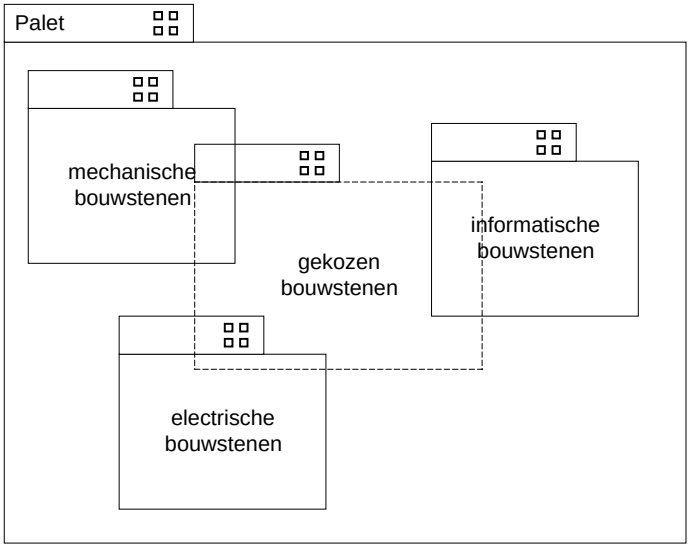
Functie is daarmee de meest eenduidige beschrijving van een systeem die mogelijk is en daarmee wellicht de belangrijkste. De *functionele structuur* en het functionele model zijn bepalend voor de vorm en ook voor de werkzaamheden en de structuren in alle andere weergaven van het systeem. De functionele structuur van een systeem is in een goed ontworpen systeem ook evenredig terug te vinden in alle andere weergaven van het systeem.

4.6 Palet

Het principe van meervoudigheid van vorm stelt dat een bepaalde functie door vele verschillende vormen kan worden geleverd. Bij het opbouwen van een systeem kan daarom ook worden gekozen uit een hele verzameling van bouwstenen. De bouwstenen van een ontwerper van elektronische schakelingen zijn de weerstanden, transistoren, draadjes en andere elektronische bouwstenen waarmee hij het systeem gaat bouwen. Voor de doe-het-zelver zijn het de spulletjes die hij kan kopen bij de bouwmarkt. En voor een ontwerper van softwaresystemen zijn het recordtypes, classes en procedures. De bouwstenen die een ontwerper of constructeur ter beschikking heeft, vormen tezamen zijn *palet*.

Het palet is de verzameling bouwstenen (de bouwdoos) die kan worden gebruikt om de functie van het systeem te realiseren.

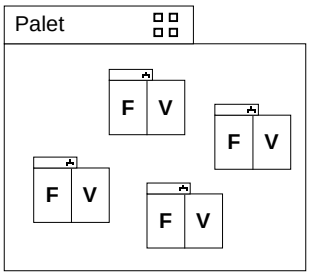




Figuur 34: het palet is de verzameling van de meest geschikte of ter beschikking staande bouwstenen

De bouwstenen in het palet hebben een tweezijdig karakter: ze hebben een bepaalde functie én ze hebben een bepaalde vorm. Daarmee vormt het palet de schakel tussen functie en vorm. Door de bouwstenen uit het palet op de juiste manier te combineren kan een systeem met de gewenste functie worden gebouwd.

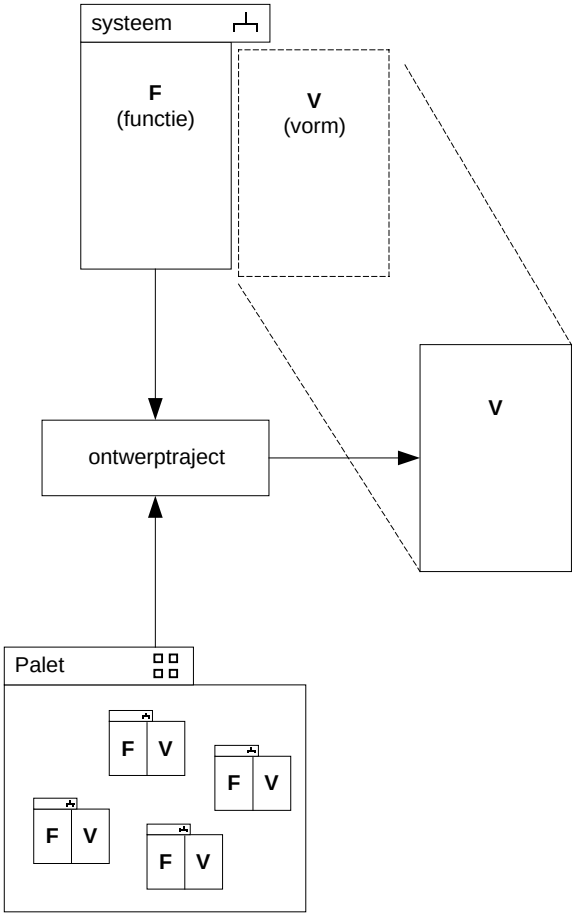
Een bouwsteen is een (klein) systeem en heeft functie én vorm, net als elk ander systeem. Het palet is daarmee de schakel tussen functie en vorm.



Figuur 35: het palet is de schakel tussen functie en vorm: elke bouwsteen heeft functie (F) én vorm (V)

4.7 Ontwerptraject

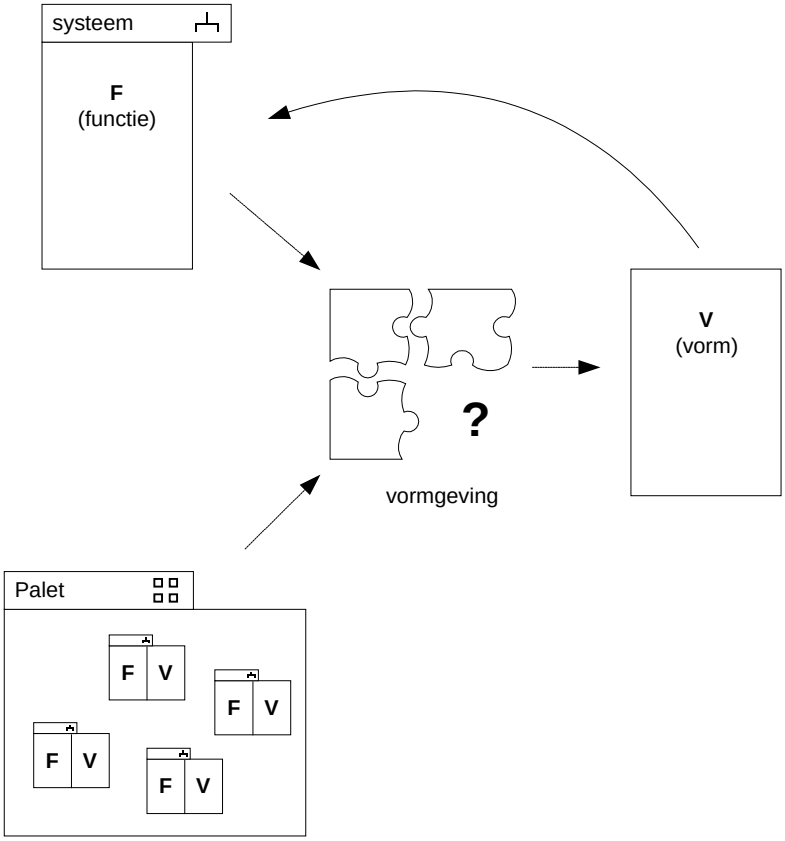
Systeemontwikkeling is een proces dat begint met een gewenste functie en een palet met bouwstenen en dat in stappen moet leiden tot een bruikbare vorm.



Figuur 36: het ontwerptraject levert een vorm bij een functie, samen 'het systeem'

Als het palet is gekozen, kan er worden geprobeerd een samenstelling van bouwstenen te bedenken (vorm) die voldoet aan de gegeven functie. Hetzij door de bouwstenen daadwerkelijk te rangschikken, hetzij door een *bouwtekening* of constructievoorschrift te maken. Soms is deze stap eenvoudig. Het bedenken van een geschikte vorm voor het op zijn plaats houden van een schilderij is eenvoudig. Een spijker, een draadje en twee haakjes kunnen de gewenste functie leveren, dat is gemakkelijk te bedenken. Soms is de stap van functie naar vorm moeilijk. Het bedenken van de geschikte vorm om de besturing van een groot gerobotiseerd logistiek centrum aan over te kunnen laten is niet gemakkelijk.

De creatieve stap die nodig is om díe combinatie van bouwstenen te bedenken die een vorm met de gewenste functie oplevert, heet *vormgeving*.



Figuur 37: bouwstenen worden door vormgeving gecombineerd tot vorm met de juiste functie

Vormgeving kan maar op twee manieren worden gedaan: menselijke inventiviteit of *evolutie* (trial-and-error). Net als puzzelen. Evolutie is echter willekeurig en langdurig, en vormgeving zal daarom meestal een creatieve bezigheid zijn. Er zal hiervoor dan moeten worden vertrouwd op menselijke inventiviteit.

> *De meest essentiële stap van het ontwerpproces, de vormgevingsstap, zal altijd zijn gebaseerd op menselijke creativiteit. Het enige alternatief is namelijk (langdurige en toevallige) evolutie.*

4.7.1 Functiemodellering

Eenvoudige systemen, systemen met een eenvoudige functie, kunnen in een enkele vormgevingsstap worden gemaakt. Minder eenvoudige systemen moeten in meerdere vormgevingsstappen worden gemaakt. Deel-voor-deel, of aspect-voor-aspect. In alle situaties geldt echter dat als het niet mogelijk is om in één keer een mentale voorstelling te maken van de juiste *compositie* van bouwstenen, er niets anders op zit dan de functie eerst te verdelen in kleinere, meer elementaire functies en het dan nog een keer te proberen. En dat net zolang tot elke functie wél kan worden vormgegeven.

De eerste stap van de ontwikkeling van een groot systeem is daarom meestal de *functionele decompositie* van het systeem.

Tijdens de functionele decompositie wordt de functie van het systeem recursief *gedecomponeerd* in steeds kleinere en dus ook steeds eenvoudiger functies. Dit *recursieve decompositieproces* moet worden volgehouden totdat de vormgevingsstap van elke individuele functie eenvoudig genoeg is geworden om 'als mentale puzzel te kunnen worden opgelost'.

Tijdens elke decompositiestap moet het eventuele palet of de gekozen realisatietechnologie goed in het oog worden gehouden, uiteindelijk moet er namelijk een functionele structuur ontstaan die goed aansluit bij het palet of de gekozen manier van realiseren. Door bij elke decompositiestap te proberen dichter bij de functie van de bouwstenen in het palet te komen, ontstaat uiteindelijk aansluiting tussen de functiestructuur en de bouwstenen. Zodra er aansluiting is gevonden, kunnen de bouwstenen op de juiste wijze worden samengenomen en is het systeem in principe te realiseren.

Hóe een functie moet worden gedecomponeerd tot een structuur van deelfuncties, zodanig dat de deelfuncties beter aansluiten op de functies van de beschikbare bouwstenen, is puzzelwerk. Net als het kiezen van beter passende bouwstenen puzzelwerk is. Net als ook het kiezen van de juiste combinatie bouwstenen voor de gegeven functie puzzelwerk is. Deze stappen kunnen niet (nooit) door een methode worden beschreven. Een methode kan hooguit zorgen voor de juiste voorwaarden. Dit is er de reden van dat nooit het hele ontwikkelproces kan worden geautomatiseerd of in een methode kan worden ondergebracht. De kunst is ervoor te zorgen dat de puzzel nooit te veel stukjes bevat…

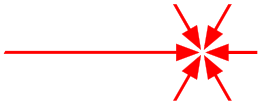
> *Als het moeilijk is om direct een vorm voor een functie te bepalen, dan moet er eerst een functionele decompositie worden gedaan. Het doel is deelfuncties te laten ontstaan die wel gemakkelijk kunnen worden vormgegeven (hetzij door daadwerkelijke compositie van bouwstenen, hetzij door het opstellen van een bouwtekening of model).*

4.7.2 Functiemodellen: verdeel en heers

Complexe problemen kunnen alleen worden opgelost door ze te verdelen in kleinere, minder complexe problemen. Los alle deelproblemen op en het oorspronkelijke probleem is ook opgelost! Voor een ontwerper van systemen is het probleem 'het bouwen van een systeem dat aan de gewenste functie kan voldoen'. Omdat er nog niets anders bestaat dan de functie van het systeem, immers, de vorm moet nog worden bepaald, is functiedecompositie zijn enige mogelijkheid om het systeem te verdelen in eenvoudigere delen. Functiedecompositie is daarmee het belangrijkste middel dat de ontwerper en constructeur van grote systemen hebben om complexiteit te beteugelen.

> *Functiedecompositie is de enige manier om het ontwerpen van systemen met complexe functionaliteit eenvoudig te kunnen houden.*

Het resultaat van een functiedecompositie is een *functiemodel*.



Een functiemodel is een onveranderlijke structuur van functies en koppelingen.

Omdat een functiemodel ook kan worden gemaakt zonder dat de wijze van construeren of realiseren van een systeem bekend is, is het een universeel model dat in alle engineeringdisciplines kan worden gebruikt. Functiedecompositie is technologieonafhankelijk.

Het functiemodel is nodig om een nog te bouwen systeem te kunnen opdelen in eenvoudigere delen. Functiemodellering is universeel: het is niet gebonden aan de wijze van realiseren van het systeem.

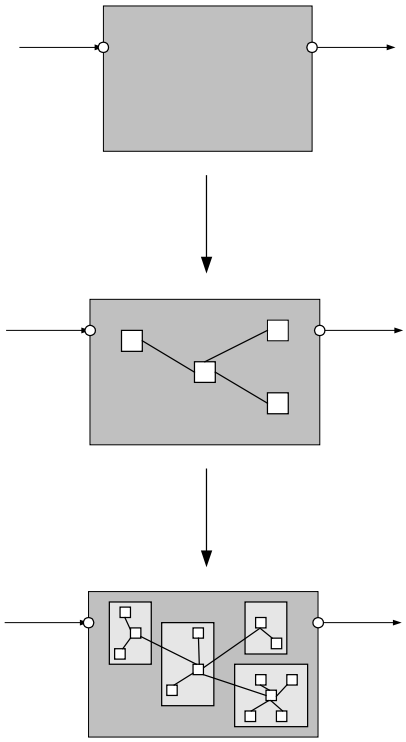
Het functiemodel is als het ware de taakverdeling binnen het systeem: ‘welk deel verricht welke taak’.

De aard van functionele decompositie is zodanig dat de resulterende (functionele) structuur van een complex systeem verschillende niveaus van aggregatie kent: functies zijn opgebouwd uit kleinere functies enzovoort. De functionele structuur van een complex systeem is dus modulair én hiërarchisch.

Een functiemodel is modulair én hiërarchisch.

Een functiemodel is een structuur die het de ontwerper mogelijk maakt de vorm van het systeem eenvoudig te kunnen ontwerpen of bouwen. Bij het maken van een functiemodel wordt een functie die te complex is om direct te kunnen worden gerealiseerd, of waarvoor niet direct een bouwtekening kan worden opgesteld, opgedeeld in deelfuncties. De ontwerper probeert in de gegeven functie de meest *dominantie deelfuncties* te vinden en hun onderlinge koppelingen te bepalen. Dat doet hij in principe op gevoel. Verderop in dit boek zullen methoden worden gegeven die hem houvast geven, maar de aanpak zal altijd wat gevoelsmatig zijn.

Functiemodellering is gebaseerd het (gevoelsmatig) bepalen van een structuur van meest dominantie deelfuncties in een gegeven functie.



Figuur 38: functiemodellering is het recursief decomponeren van de meest dominante deelfuncties totdat een eenvoudig realiseerbare structuur is ontstaan

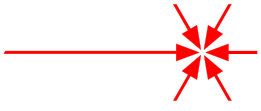
In principe is elke structuur van deelfuncties geschikt om te dienen als uitgangspunt voor verdere modellering of voor realisatie, zolang de structuur maar een goede modulariteit heeft (duidelijke modules, met hoge cohesie en lage koppeling). Als de ontwerper een functiestructuur heeft bedacht die niet voldoet aan dit criterium, dan zal hij een stap terug moeten doen en een andere opdeling moeten bedenken.

Systemen die deel uitmaken van een groter geheel (en dat zijn bijvoorbeeld productiebesturingssystemen), moeten een functiestructuur krijgen die aansluit bij dat grotere geheel of *supersysteem*. Later zal daar in detail op worden ingegaan.

> *Systemen die nauw moeten aansluiten op een groter geheel of supersysteem, moeten een functiestructuur krijgen die evenredig is aan die van het supersysteem.*

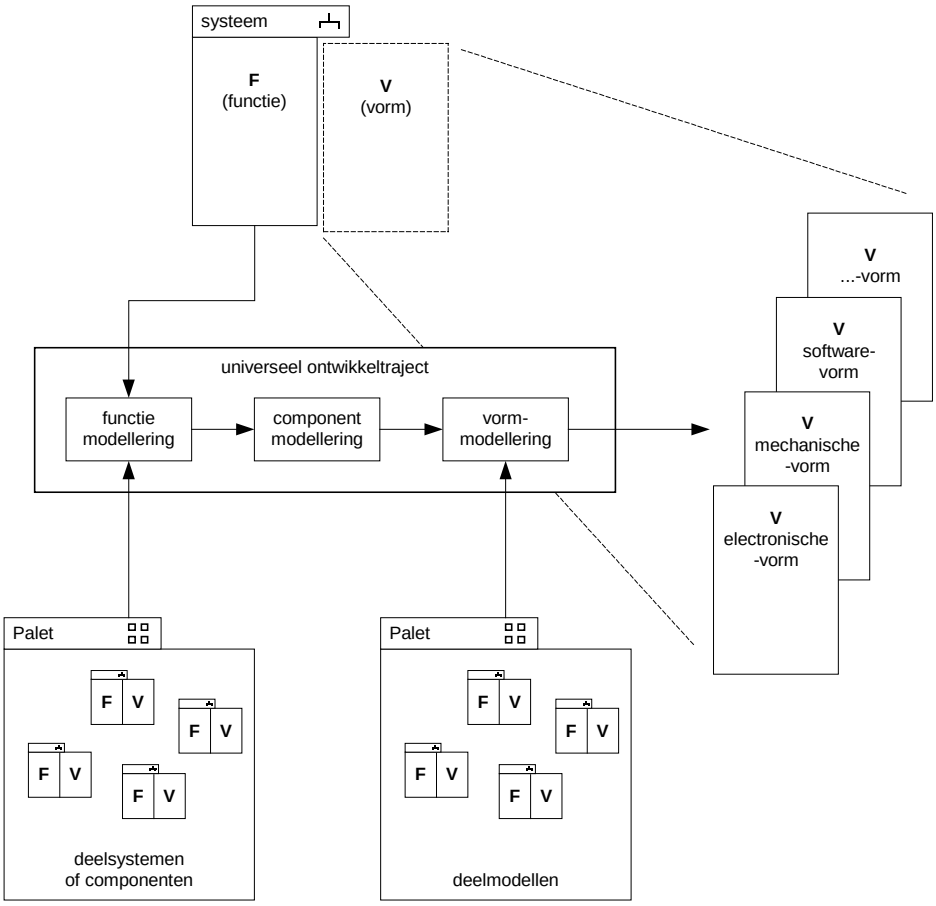
4.7.3 Vormmodellering

Aansluitend aan de functiemodellering volgt de vormmodellering. In deze fase worden de bouwtekeningen of *vormmodellen* ontwikkeld. Elk engineeringdiscipline heeft zijn eigen wijze van vormmodelleren. De bouwtekeningen van een torenflat zijn net zo zeer een vormmodel als de datamodellen van een grote database.



4.7.4 Componentmodellering

Tijdens de functiemodellering wordt er van het systeem een soort van ‘verdeel-en-heers-kaart’ gemaakt. Elke individuele functie van het model kan later als bouwsteen worden herkend in het systeem. Om ervoor te zorgen dat de gewenste functionele structuur behouden blijft tijdens de *vormmodelleringfase*, wordt elke functie voordat deze fase begint alvast toegekend aan een structuur van *componenten*. De componenten vormen de toekomstige behuizingen van bouwstenen, in de ontwerpfase nog weergegeven op bouwtekeningen. Per component wordt een *bouwtekening* opgesteld. Het universele *ontwikkeltraject* bevat dus drie stappen: functie-, component- en vormmodellering. Onderstaande figuur toont dit.



Figuur 39: functiemodellering en vormmodellering worden verbonden door component-modellering

4.7.5 Functionele evenredigheid

De reden om een functiemodel te maken is ‘complexiteit’. Een functiemodel is zo gekozen dat nog te bouwen of reeds bestaande systemen zo eenvoudig mogelijk kunnen worden onderhouden, samengesteld of anderszins behandeld. Het functiemodel heeft daarvoor het systeem verdeeld in niet alleen eenvoudige, maar ook ‘handige’ delen. Daarom is het belangrijk dat alle andere weergaven, modellen en aspecten van het systeem een structuur hebben die evenredig is met de functionele structuur van het systeem. Er is geen reden te bedenken die rechtvaardigt het systeem eerst door functiemodellering te verdelen in eenvoudigere delen en dan vervolgens tijdens het maken van bijvoorbeeld de bouwtekeningen die eenvoudige structuur niet te respecteren: de vormstructuur van een complex systeem is dus óók modulair én hiërarchisch en bovendien *functioneel evenredig*.

> *Alle modellen, deel- en aspectsystemen van het systeem moeten evenredig zijn aan het functiemodel.*

4.8 Samenvatting

Functie is de 'bruikbaarheid' van datzelfde systeem. *Vorm* is de drager van functie. Vorm en functie zijn beide concreet, dus eigenschappen van het werkelijke systeem.

Functie is niet bepalend voor vorm. Het *principe van onbepaaldheid van vorm* stelt dat functie geen uitspraak doet over de vorm van een systeem.

Voor een bepaalde functie zijn ook meerdere vormen te bedenken en een bepaalde vorm kan worden veranderd zonder dat zijn functie verandert. Dit wordt beschreven door *het principe van meervoudigheid van vorm*.

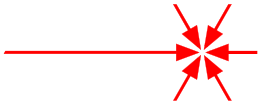
Om een functie te kunnen *vormgeven* moet ook bekend zijn welke bouwstenen daarvoor kunnen worden gebruikt. Deze vormen het palet. Vormgeven is dan: bouwstenen in het palet zo combineren dat ze de gewenste functie kunnen leveren. Als dit niet gemakkelijk is, dan moet er eerst een functionele decompositie plaatsvinden, net zolang tot er functies ontstaan waarvoor wel gemakkelijk een samenstelling van bouwstenen is te bedenken. Dat is het geval als de deelfuncties die volgen uit de functionele decompositie makkelijk zijn af te beelden op de functies van de bouwstenen.

Alle deelvormen die nodig zijn om een bepaalde functie te leveren zijn bestaansafhankelijk van die functie. Bestaansafhankelijkheid is de natuurlijke lijm die de structuur van systemen bepaalt.

Goede systemen en modules hebben een hoge cohesie. Dat wil zeggen dat alle delen bestaansafhankelijk zijn van de functie van het geheel.

Een functiemodel is de enige mogelijkheid die een ontwerper heeft om complexe systemen in eenvoudige delen op te delen zonder dat daarvoor de concrete vorm van het systeem bekend moet zijn.

Het opstellen van een functiemodel is gebaseerd op het vinden van de meest dominante deelfuncties en hun koppelingen. Elk functiemodel dat een goede



modulariteit en hiërarchie heeft, kan worden overwogen als basis voor realisatie of verdere modellering. Het moet daarvoor worden getest tegen de eisen die aan het systeem worden gesteld.

5. Abstractie en typering

'One should not increase, beyond what is necessary, the number of entities required to explain anything.'

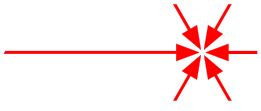
Occam's Razor (Principe van eenvoud), William of Occam (Ockham)

Hoofdstukwijzer

Onderwerp	*Paragraaf*
Waarom is abstractie alleen niet voldoende? Waarom is er nog een modelleringtechniek nodig?	Veranderlijke structuur
Bouwtekeningen hebben een onveranderlijke structuur, hoe kunnen we dan veranderlijke structuren toch weergaven met een onveranderlijk model?	Typeren
Kan worden beredeneerd in termen van structurele complexiteit hoe abstractie en typering werken?	Vereenvoudiging
Hoe noemen we de modules van een abstraherend model en een typerend model in een context waar beide naast elkaar bestaan?	Instanties en types

5.1 Veranderlijke structuur

Een systeem kan worden ontworpen door het eerst vorm te geven in het abstracte structuurdomein. Het abstracte structuurdomein bevat de abstracte weergaven ofwel modellen van het systeem. Een voorbeeld is de bouwtekening van een huis. Of een schaalmodel van een auto. Het abstracte structuurdomein alléén is echter niet voldoende, er bestaat namelijk een klasse van systemen die niet gemakkelijk te modelleren is in het ‘normale’ abstracte domein. Dit is de klasse van systemen met veranderlijke structuur. Van een veranderlijke structuur kan namelijk geen bouwtekening worden gemaakt. Welk moment in het bestaan van de structuur moet worden gekozen als de structuur op elk moment anders kan zijn? ‘Normaal’ abstraheren is dan niet voldoende. In modelleringopzicht kan wat dat betreft onderscheid gemaakt worden tussen twee soorten structuren:



- Veranderlijke structuren

  Een systeem heeft een veranderlijke structuur als de herhaling, diversiteit of koppeling van de structuur tijdens de levensduur ervan verandert.

  De structuur van een bepaalde familie (bijvoorbeeld de familie Jansen) is veranderlijk in de tijd. Er kunnen kinderen en ouders bij komen of weg gaan.

- Onveranderlijke structuren

  Een systeem heeft een onveranderlijke structuur als de herhaling, diversiteit en koppeling van de structuur gedurende de levensduur ervan gelijk blijven.

  De structuur van een hangbrug is onveranderlijk. De structuur van een fiets ook, ondanks de bewegende delen, er komen geen onderdelen of verbindingen bij.

> *Een structuur is veranderlijk als tijdens de levensduur van de structuur de herhaling, diversiteit of koppeling van modules verandert.*

Als een structuur veranderlijk is, kan het wenselijk zijn om met een meer statische weergave van het systeem te kunnen werken. Dit kan worden bereikt met *typerend model*. Voor veel systemen met een veranderlijke structuur is de onderliggende *karakteristieke structuur* namelijk wél onveranderlijk. Deze onderliggende karakteristieke structuur kan dan in een model worden geplaatst. De structuur van zo'n typerend model is representatief voor elke mogelijke structuur die het systeem in zijn levensduur kan aannemen. Het achterhalen van de karakteristieke, statische structuur van een systeem heet *typeren*.

> *De typering van een veranderlijke structuur heeft zelf een onveranderlijke structuur. Veranderlijke (en onveranderlijke) systemen hebben onveranderlijke typerende modellen.*

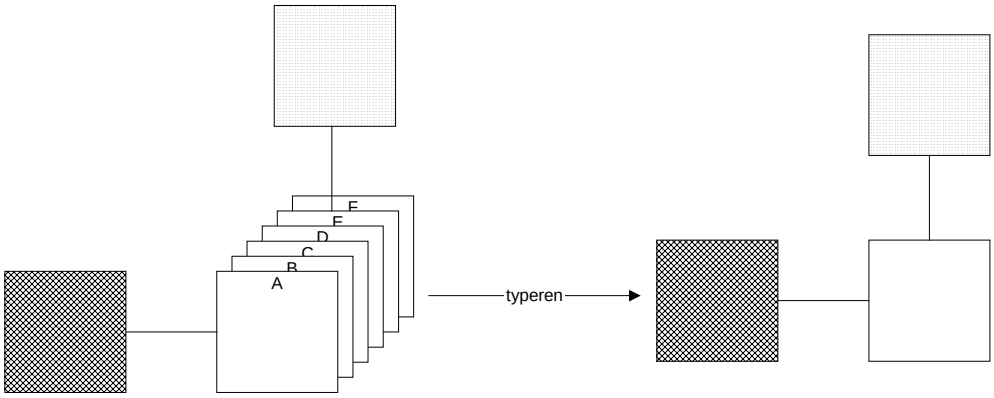
Een voorbeeld is het verschil tussen de functiestructuur van een systeem en de vormstructuur van een systeem: de functiestructuur is onveranderlijk terwijl de vormstructuur dat niet hoeft te zijn. Een systeem verandert niet van functionaliteit of van structuur van functie, maar veel systemen hebben wel een veranderlijke structuur van vorm nodig om die functie te kunnen leveren. De structuur van de cellen die samen de nieren van een mens vormen is veranderlijk, maar de functionele structuur waarin ze zijn opgenomen niet (de structuur van organen).

> *De functiestructuur van een systeem is meestal onveranderlijk, de vormstructuur hoeft dat niet te zijn.*

5.2 Typeren

Typeren is het cumuleren van alle mogelijke structuren die een veranderlijk systeem kan aannemen. Typeren van een bestaand systeem gebeurt door modules te *groeperen* en de eigenschappen van gelijksoortige modules en structuren (herhaling) te verzamelen in een enkele karakteristieke module. Dit heet *analytisch typeren*. Wat zo ontstaat is een *type* dat representatief is voor alle *individuen* van dat type. Een type geeft de mogelijkheden van de individuen aan, níet hoe ze op een bepaald moment in hun leven die mogelijkheden benutten. Of elk individu ook daadwerkelijk al die

mogelijkheden zal benutten, hangt af van de context waarin de individuen worden gebruikt.



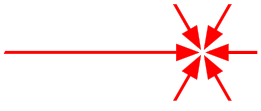
Figuur 40: typering is het samennemen van herhaling en het optellen van exemplarische mogelijkheden

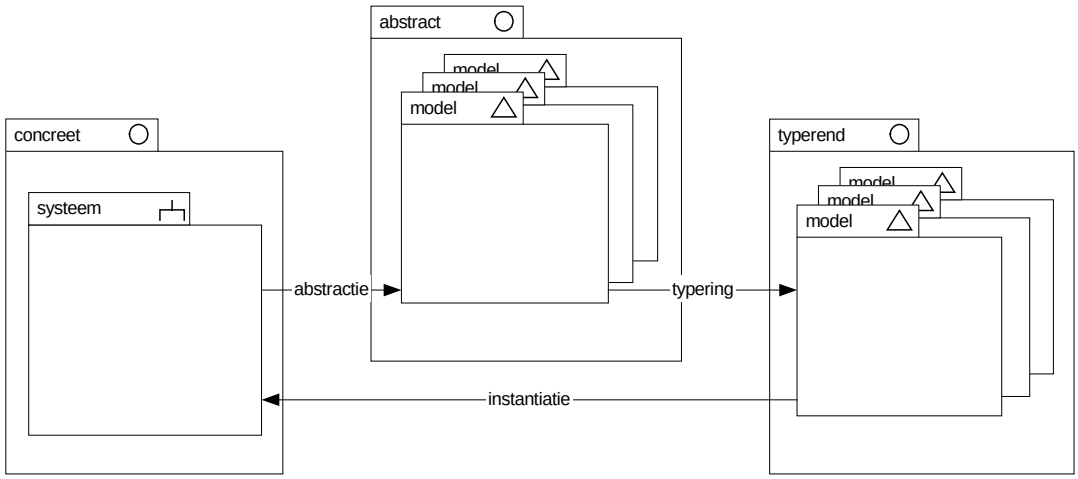
Andersom kan een type ook expliciet worden samengesteld zonder te kijken naar een bestaand systeem. Het heet dan *synthetisch typeren*. Dit gebeurt door mogelijkheden aan het type toe te kennen en dan vervolgens de individuele modules van een systeem te proberen te *classificeren* volgens de beschikbare types.

> *Typeren is het achterhalen (analytisch) of aanbrengen (synthetisch) van de karakteristieke structuur van een systeem. Een type geeft alle mogelijkheden van de exemplaren van dat type aan.*

Veranderlijke systemen (systemen met een veranderlijke structuur) blijken onveranderlijke typeringen (modellen met een onveranderlijke structuur) te hebben.

Net als abstracte modellen een eigen structuurdomein hebben waarin specifieke regels gelden, zo bestaan typerende modellen ook in hun eigen structuurdomein. In het typische domein gelden andere regels dan in het abstracte domein. Zo kan er in het typische domein geen herhaling bestaan en is de factor tijd afwezig. De verschillende structuurdomeinen zijn wel aan elkaar gerelateerd, ze worden gescheiden door transformaties als abstractie, typering en instantiatie. Via die transformaties kan een structuur in het éne domein worden getransformeerd tot een structuur in het andere domein. De ontwerper heeft er met het typische domein en de typerende weergaven als het ware een nieuwe werkruimte bij gekregen. Soms kunnen handelingen het beste worden verricht in het abstracte domein, soms in het typische domein, maar altijd moeten de weergaven in beide structuurdomeinen consistent worden gehouden. Wederom door via de tussenliggende transformaties van het éne domein naar het andere te gaan.

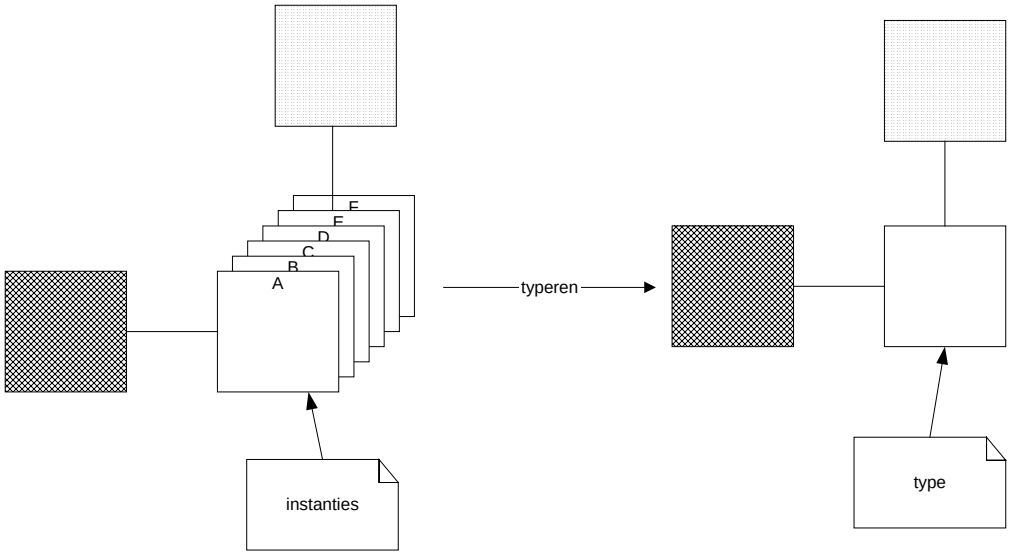




Figuur 41: naast het concrete en het abstracte structuurdomein bestaat het typische of typerende domein

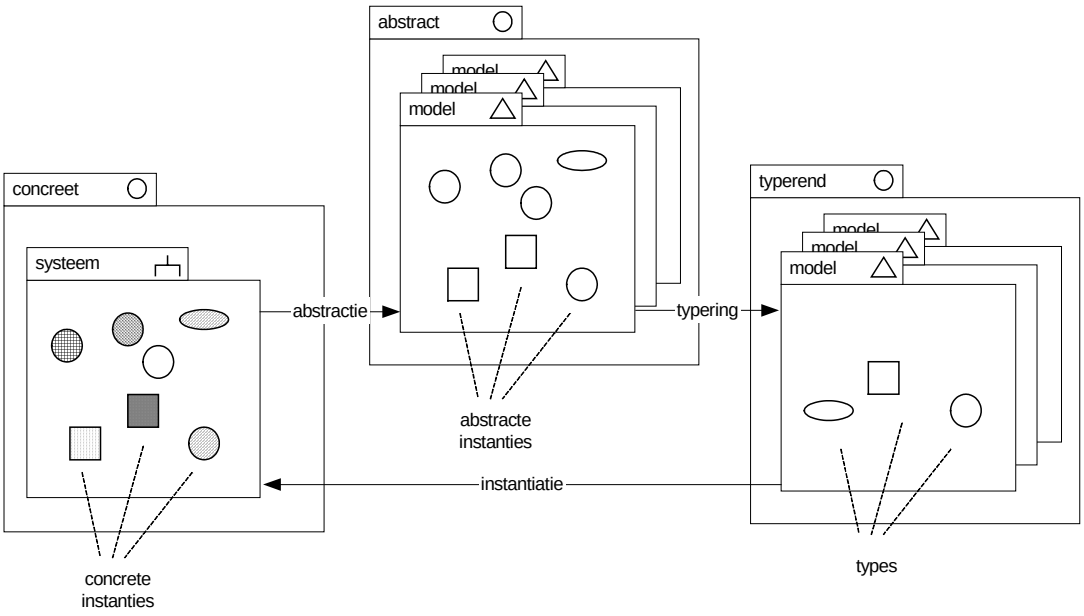
5.3 Instanties en types

Als er met typerende modellen wordt gewerkt, dan is het belangrijk om goed onderscheid te blijven maken tussen de individuele modules van het systeem of het bijbehorende abstracte model en de typerende modules van het model van dat systeem. De individuele modules waaruit een systeem en het abstracte model bestaan, worden *instanties* of *exemplaren* genoemd. De typerende modules waaruit het typerende model bestaat, worden *types* genoemd. Een type is altijd een beschrijving, de concrete of abstracte modules die voldoen aan die beschrijving zijn de instanties van dat type.

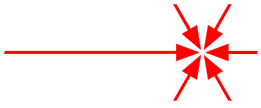


Figuur 42: instantie en type

Een typerend model bestaat in dus uit types en het door dat model weergegeven systeem of abstracte model uit instanties. Altijd als er over instanties en types wordt gesproken, komen beide in dezelfde context voor. Het is niet zinvol om over types of instanties te spreken in een context waar niet beide soorten modellen tegelijkertijd bestaan.



Figuur 43: instanties bestaan in het concrete en abstracte domein, types in het typerende domein



5.3.1 Is een bouwtekening van een auto nu een typerend model of een abstraherend model?

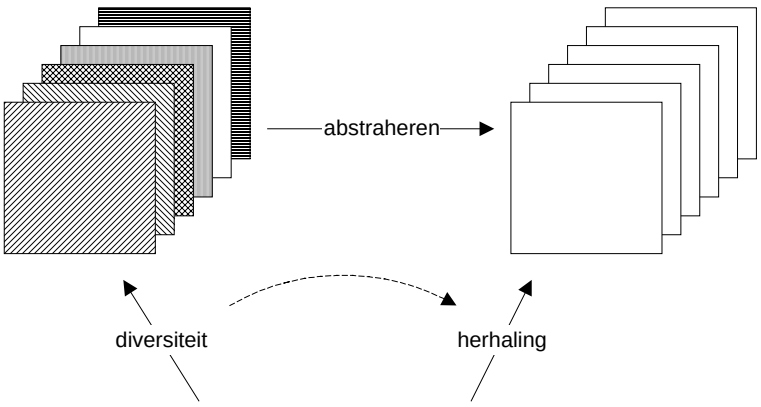
Aan de éne kant kan worden beredeneerd dat alle exemplaren van de auto worden getypeerd door hun bouwtekening en dat de bouwtekening dus een typerend model is. Aan de andere kant is ook gemakkelijk te zien dat deze verhouding van een andere orde is dan de verhouding tussen de typerende en concrete structuur van een familie. De typerende structuur van de familie houdt rekening met het feit dat een man en een vrouw meerdere kinderen, zowel meisjes als jongens, kúnnen hebben. Een concrete familie hoeft niet eens kinderen te hebben. Of maar één. En hoeft zelfs niet eens uit minimaal een man en een vrouw te bestaan. Abstractie en typering en abstracte en typerende modellen liggen dicht bij elkaar maar kunnen uit elkaar worden gehaald door de eigenschappen van de exemplaren te vergelijken met die van hun type. Als een model een onveranderlijke structuur weergeeft, dan is het niet zinnig om over typering te spreken. Hoewel er dus tussen de deegvorm en de koekjes een type / exemplaar-relatie bestaat, is het niet zinnig om dat ook op die manier te benoemen. Het voegt niets toe, behalve misschien verwarring. Pas als het type een opsomming is van exemplarische mogelijkheden en het typerende model een veranderlijke structuur weergeeft zal er over typering worden gesproken.

Als alle exemplaren van een bouwtekening gewoon eenvoudige en gelijkwaardige kopieën zijn, wordt er niet gesproken over typering. Er is dan sprake van' eenvoudige' replicatie omdat de bouwtekening geen typerend model maar een abstract model is.

5.4 Vereenvoudiging

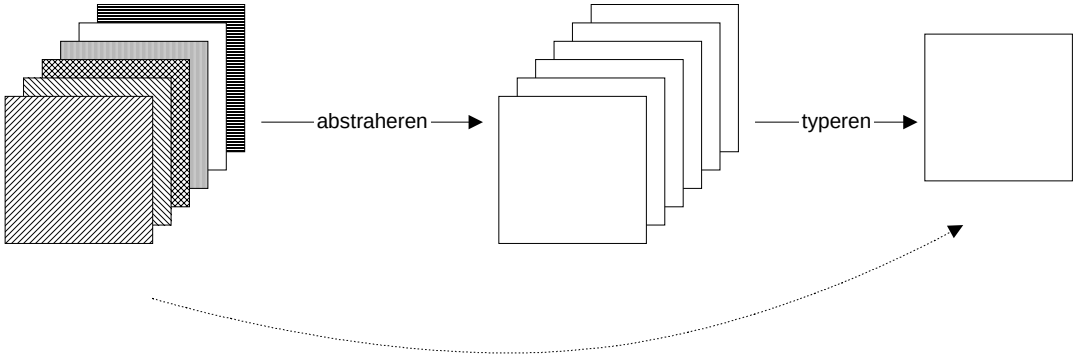
Abstractie en typering zijn vereenvoudigingstechnieken. Door te abstraheren wordt er informatie buiten het model gehouden. Door te abstraheren vermindert ook de diversiteit en vermeerdert de regelmaat en herhaling in de structuur wordt vergroot.

Abstractie vermindert diversiteit en vermeerdert herhaling in een structuur.



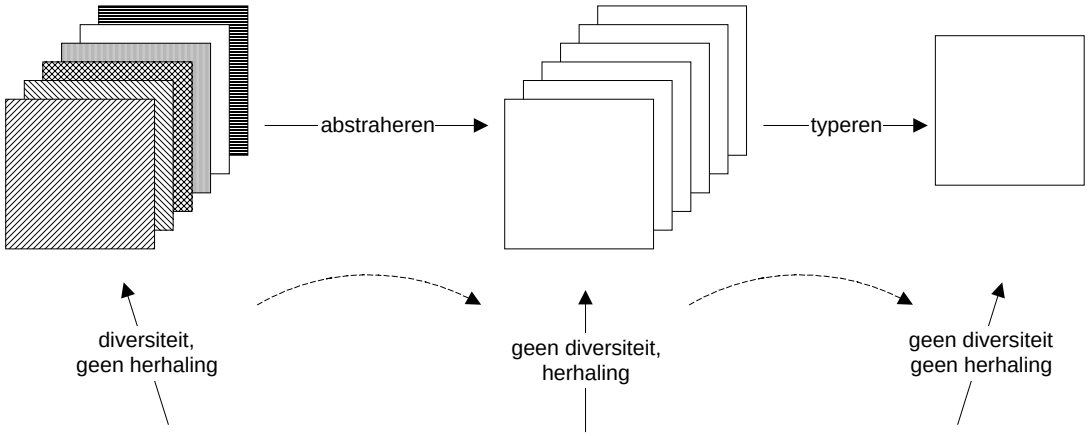
Figuur 44: abstractie vermindert diversiteit maar vergroot herhaling

Typering wordt altijd vooraf gegaan door abstractie, zonder abstractie kan er niet worden getypeerd. Zonder abstractie is alles anders, pas na het weglaten van de juiste details blijkt ineens dat er meerdere modules van dezelfde soort bestaan. Immers, als we van een rode en een blauwe bol de verschillen in kleur negeren, zijn het ineens twee (dezelfde) bollen. En dit gegeven kan weer worden gebruikt voor het *typeren* van de structuur omdat daarbij de aanwezigheid van herhaling gunstig is.



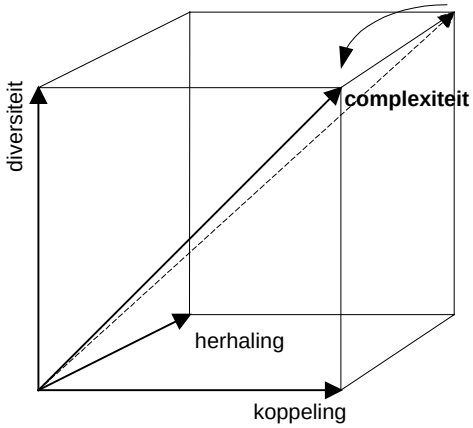
Figuur 45: zonder abstractie is geen typering mogelijk

De onderstaande figuur toont hoe herhaling pas na abstractie kan worden verwijderd.



Figuur 46: typering vermindert herhaling

In een typerend model komt veel minder herhaling voor dan in een abstract model of het systeem. Herhaling is nodig in het systeem, het hoort namelijk gewoon tot de structuur van het systeem, maar het is overbodig in de karakterisering van het systeem.



Figuur 47: een typerend model is minder complex dan een abstract want er komt (bijna) geen herhaling meer in voor

Typering vermindert de herhaling in een structuur.

Typeren kan ook met temporele structuren. Veel programmeertalen hebben daar zelfs gewoon syntaxis voor (bijvoorbeeld de `for`-lus en de `while`-lus). Om van huis naar school te lopen moet u een linker en een rechterstap nemen, en dat herhalen, net zolang totdat u bent waar u zijn wilt. Om van school naar het dagverblijf te lopen geldt weer hetzelfde, net als voor elke andere afstand die u lopend wilt afleggen. (Dit is natuurlijk de reden dat onze voorouders het wiel hebben uitgevonden.) Herhaling heeft altijd dezelfde karakteristieken en kan voor het typeren van een systeem buiten het typerende model worden gelaten.

5.5 Samenvatting

Er bestaan onveranderlijke en veranderlijke structuren. Een veranderlijke structuur heeft veranderende factoren van complexiteit (diversiteit, herhaling en koppeling).

Om van een systeem met een veranderlijke structuur een model met een onveranderlijke structuur te kunnen maken, is een speciale modelleringtechniek nodig, alleen abstractie volstaat dan niet.

Er bestaan twee basistechnieken om te kunnen vereenvoudigen (analyse) of modelleren (synthese):

Vereenvoudigingstechnieken		
Abstractie	Hoe	Abstractie is het bewust negeren van irrelevante informatie.

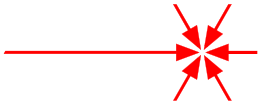
Vereenvoudigingstechnieken		
	Waarom	Pas abstractie toe om een model te ontdoen van informatie die voor het doel van het model onnodig of irrelevant is zodat het model eenvoudiger is dan het systeem.
Typering	Hoe	Typeren is het cumuleren van alle mogelijke toestanden van een structuur tot een karakteriserende structuur.
	Waarom	Pas typering in een model toe als het systeem een veranderlijke structuur heeft en de karakteristieke structuur daarvan zichtbaar moet worden gemaakt.

In een context waar zowel een abstraherend als een typerend model wordt gebruikt, moet onderscheid worden gemaakt tussen de instanties van het abstraherende model en de types van het typerende model.

In een context zonder typerende modellen heeft het geen zin over instanties te praten.

De twee soorten modellen hebben allebei een eigen structuurdomein: het abstracte domein en het typische domein. In elk structuurdomein gelden andere regels. In het abstracte domein bestaan alleen instanties, geen types. In het typische domein bestaan alleen maar types, geen instanties.

Vereenvoudiging is onomkeerbaar. Vanuit een model kan nooit meer het eventuele origineel worden achterhaald.



6. Herstructurering

Hoofdstukwijzer

Onderwerp	*Paragraaf*
Als er meerdere vormen met dezelfde functie zijn te bedenken, is het dan ook mogelijk om een gegeven vorm van structuur te veranderen zonder dat zijn functie verandert?	Herstructurering van structuur
Waarom is de structurele complexiteit van een structuur hetzelfde voor en ná herstructurering?	Herhaling versus koppeling
Is het mogelijk om zonder dat de werking van de structuur verandert een betere structuur te verkrijgen door de modulesamenstelling in een structuur te veranderen?	Factorisatie
Is het mogelijk om zonder dat de werking van de structuur verandert een betere structuur te verkrijgen door afhankelijkheden te verleggen en modules te parameteriseren?	Parameterisatie
Wat is het fundamentele mechanisme dat schuilt achter alle vormen van herstructurering?	Symmetrie en compressie
Als de vormgeving van een structuur kan worden veranderd zonder dat zijn functie verandert, is het dan ook mogelijk de weergave of notatie van die structuur (het model) te wijzigen zonder dat het model een ander systeem gaat weergeven?	Abstracte en concrete herstructurering

6.1 Herstructurering van structuur

Volgens het principe van meervoudigheid van vorm zijn er meerdere vormen mogelijk voor een gegeven functie. Daarom moet het ook mogelijk zijn om de vorm van een systeem te veranderen zonder dat de functie van het systeem verandert. De techniek waarmee de structuur van een geheel kan worden veranderd zonder dat de functie van het geheel verandert, heet *herstructureren*. Herstructurering van een geheel is het

opnieuw rangschikken van de delen van het geheel. Bijvoorbeeld minder herhaling van delen en dus betere veranderbaarheid. Of juist meer herhaling van delen en minder koppelingen, en daardoor betere herbruikbaarheid of robuustheid.

> *Herstructurering is het veranderen van een structuur van een geheel zonder dat de functie van het geheel verandert.*

Er zijn maar enkele manieren om te herstructureren. Allereerst is het mogelijk om de samenstelling van modules te veranderen (herverdelen van delen over de gehelen). Bijvoorbeeld zodanig dat er zo min mogelijk koppeling bestaat tussen de modules. Dat kan door modules die door meerdere andere modules worden gebruikt, te herhalen in die betreffende modules. Het systeem wordt hierdoor groter, maar de modules hebben minder onderlinge koppelingen nodig. Ze worden zelfstandiger.

Een andere mogelijkheid is het onderling delen van modules. In dat geval wordt de gedeelde of gelijke functionaliteit van twee modules in een aparte, gedeelde module ondergebracht. Dit levert een compactere structuur op die vaak beter aanpasbaar is. Elke aanpassing hoeft in het ideale geval maar op één plaats in de structuur te worden aangebracht. Goede redenen om op deze manier te herstructureren zijn aanpasbaarheid en het bewaren van *consistentie* bij wijzigingen. Herstructurering door het herhalen of delen van modules heet *factoriseren*.

Naast herhalen of delen, wat voornamelijk het slim verdelen van modules betreft, is er nog een techniek om een structuur te kunnen wijzigen zonder verandering van functie. Afhankelijkheden die modules hebben, kunnen namelijk worden verwijderd door ze niet zelf de dingen die ze nodig hebben te laten ophalen (via een koppeling), maar deze als parameters mee te geven bij de opdrachten die aan de modules worden gegeven. Dit heet *parameteriseren*.

Parameterisatie ontkoppelt de betreffende module maar verlegt de koppeling naar een andere module (de aansturende). Parameterisatie is een hulpmiddel om afhankelijkheden van modules naar hun omgeving te ‘verschuiven’, zodat de modules zelf onafhankelijk zijn van hun omgeving.

Herstructurering is het toepassen van één van de volgende fundamentele *herstructureringstechnieken*:

- Factoriseren

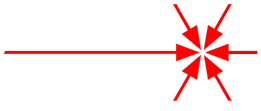
  Wat is de juiste clustering van onderdelen? Kan een structuur met minder herhaling worden geformuleerd zodat deze gemakkelijk kan worden aangepast? Kunnen sommige delen van de structuur zelfstandiger worden gemaakt door juist herhaling aan te brengen?

- Parameteriseren

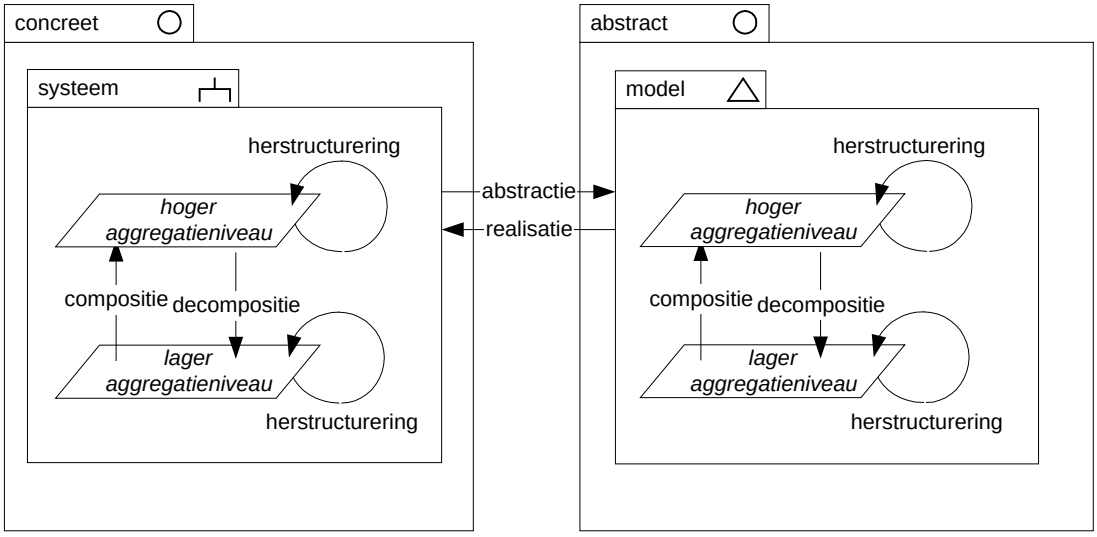
  Kunnen de modules van een structuur zelfstandiger worden gemaakt door ze contextonafhankelijk te maken? Kan een structuur compacter worden geformuleerd door sommige onderdelen generieker te maken?

> *Factorisatie en parameterisatie zijn herstructureringstechnieken.*

Herstructureringstechnieken veranderen niet het aggregatieniveau van een structuur. En net als structurering (decompositie en compositie) is herstructurering gebonden aan



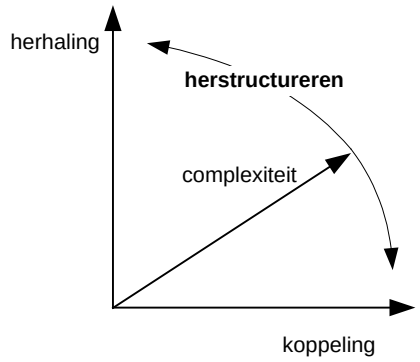
een bepaalde weergave van het systeem. Door te herstructureren ontstaat niet ineens een ander soort weergave, alleen een andere structuur in dezelfde weergave. Herstructurering kan op elk niveau van aggregatie plaatsvinden.



Figuur 48: herstructurering is gebonden aan een bepaald aggregatieniveau

6.2 Herhaling versus koppeling

Door te herstructureren kan koppeling worden verruild voor herhaling en, andersom, herhaling voor koppeling. Herhaling en koppeling zijn factoren van structurele complexiteit en een structuur kan zonder verandering van functie worden gebalanceerd naar de optimale verhouding tussen deze twee factoren.



Figuur 49: factoriseren is het balanceren van factoren van structurele complexiteit

Met wat fantasie kan worden gesteld dat complexiteit een vector is en dat de richting van die vector kan worden veranderd door herstructurering. Niet de lengte, alleen de

richting. De complexiteit blijft, alleen de verdeling ervan over de factoren herhaling en koppeling verandert.

> *Herstructurering vermindert niet de complexiteit van een structuur maar verandert de verdeling ervan over factoren.*

Wat de optimale verhouding is, is afhankelijk van de eisen die worden gesteld aan de structuur. In structuren met veel herhaling is het moeilijker consistentie te bewaren bij wijzigingen. Wijzigingen moeten door de herhaling van delen vaak op meer plaatsen in de structuur worden aangebracht. Structuren met veel herhaling zijn ook groter, omvangrijker. De voordelen van herhaling zijn echter betere modulariteit en grotere robuustheid (het falen van een module heeft veel beperktere gevolgen). Door koppeling op de juiste plaatsen te ruilen voor herhaling kunnen delen zelfstandiger worden gemaakt. In een structuur waarin herhaling voor komt, is de kans dat een falend deel het gehele systeem beïnvloedt veel kleiner.

6.3 Factorisatie

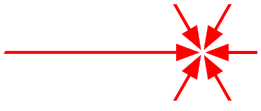
Met wat wiskundige beeldspraak zou factoriseren kunnen worden gezien als het spelen met de haakjes in een expressie. Buiten haakjes brengen, binnen haakjes brengen of uitschrijven, in ieder geval het herformuleren van een expressie zodanig dat deze het meest leesbaar of bruikbaar is. Dit kan op verschillende manieren. Een voorbeeld uit de wiskunde:

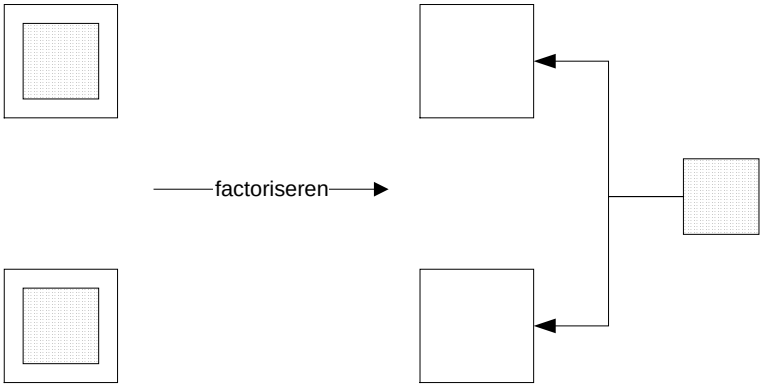
$$AB + AC \Rightarrow A\ (\ B + C\ )$$

> *Factorisatie is het herstructureren van delen van een geheel zonder dat de functie van het geheel verandert.*

Informeel kan worden gesteld dat factoriseren het herschrijven van een expressie is, zonder dat de uitkomst van de expressie verandert.

Factorisatie kan bijvoorbeeld worden gebruikt om herhaling uit een structuur te verwijderen. Op deze wijze wordt het intensief toegepast in de databasetechnologie, namelijk op de structuur van gegevensverzamelingen. In dat geval heet het datanormalisatie en gebeurt het aan de hand van normalisatieregels, maar het principe is hetzelfde. Door een gezamenlijke factor buiten haakjes te brengen, wordt herhaling verminderd ten koste van extra koppeling.



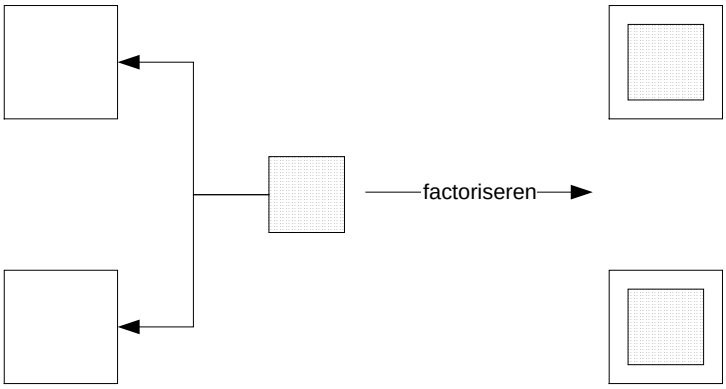


Figuur 50: factorisatie 'ruilt' herhaling tegen koppeling (of andersom)

Deze vorm van factorisatie (het verwijderen van herhaling) is voordelig als er regelmatig wijzigingen aan de inhoud van de structuur worden verwacht. De gefactoriseerde versie van de structuur kan eenvoudiger worden aangepast omdat wijzigingen maar op één plaats moeten worden aangebracht. Deze vorm van factorisatie maakt de structuur kleiner en compacter en maakt het gemakkelijker om consistentie te bewaren bij wijzigingen. Het nadeel van het verwijderen is echter dat er een ingewikkelder structuur ontstaat, met meer koppelingen.

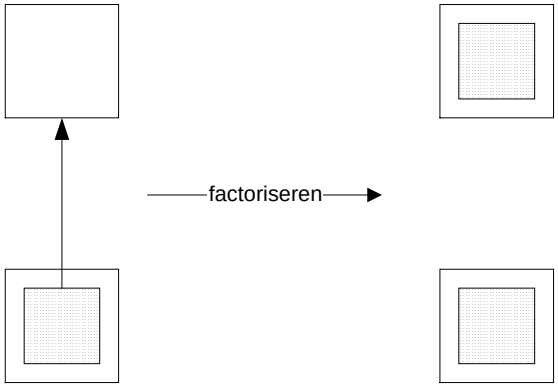
Herhaling kan worden vervangen door koppeling.

Een andere manier om een structuur te herformuleren is juist door herhaling aan te brengen. Dit vermindert weer de koppeling. Dit is gewoon de omgekeerde handeling. Ook dit is een vorm van factorisatie.



Figuur 51: factorisatie 'ruilt' koppeling tegen herhaling

De onderstaande figuur toont een structuur waarin door herhaling van deelmodules een betere ontkoppeling ontstaat. Na toepassing van deze specifieke factorisatie is er geen koppeling meer tussen de beide modules.



Figuur 52: factoriseren kan koppeling verwijderen en zelfstandigheid van onderdelen bevorderen

Een voorbeeld in termen van wiskundige expressies, eerst de uitgangssituatie (A gebruikt B):

```
A = B + 2
B = 3
```

Dan de situatie met herhaling (A en B zijn onafhankelijk):

```
A = 3 + 2
B = 3
```

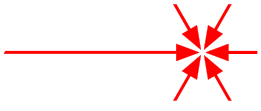
A kan nu worden gebruikt zonder B. Mocht B veranderen, dan moeten er nu echter twee expressies worden gewijzigd.

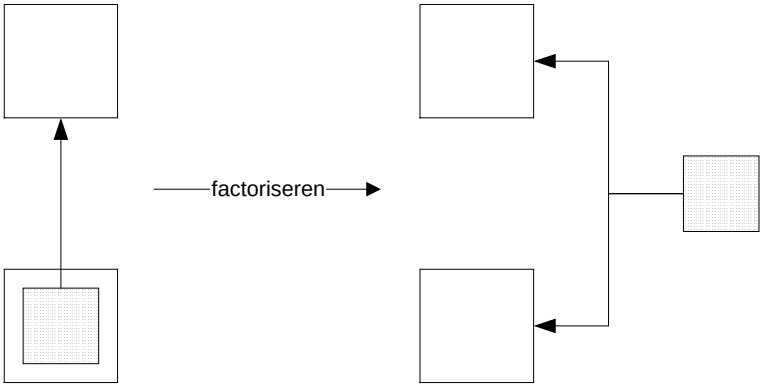
Deze vorm van factorisatie kan gewenst zijn als beide supermodules een eigen, onafhankelijke levenscyclus moeten hebben, of als de koppeling die er oorspronkelijk was, ‘erg interlokaal’ was (met andere woorden: twee niet naastliggende delen van een structuur met elkaar verbond).

Elke module, of het nu een bedrijfsafdeling, een C++-class, een operatie of wat dan ook is, kan worden ontkoppeld van een andere module door het aanbrengen van herhaling. Na herhaling van dat deel van de module dat de koppeling veroorzaakte, is de koppeling tussen de oorspronkelijk gekoppelde modules verdwenen.

Koppeling kan worden vervangen door herhaling.

Behalve het verwijderen van een koppeling door herhaling, kan een koppeling ook worden verlegd. Ook dit kan worden bereikt door factorisatie. Bij deze vorm van factorisatie wordt dat deel van een module dat de koppeling veroorzaakt apart geplaatst. De koppeling met die deelmodule blijft, maar de koppeling tussen de twee oorspronkelijke modules is verdwenen, soms is dat voordeliger. Deze oplossing heeft als voordeel boven het herhalen van deelsystemen dat er geen herhaling optreedt. Het nadeel is dat er meer diversiteit en meer koppeling ontstaat, de oplossing is dus in principe niet eenvoudiger (of complexer).





Figuur 53: factorisatie kan modules ontkoppelen

Een voorbeeld in termen van wiskundige formules, eerst de uitgangssituatie (A gebruikt B):

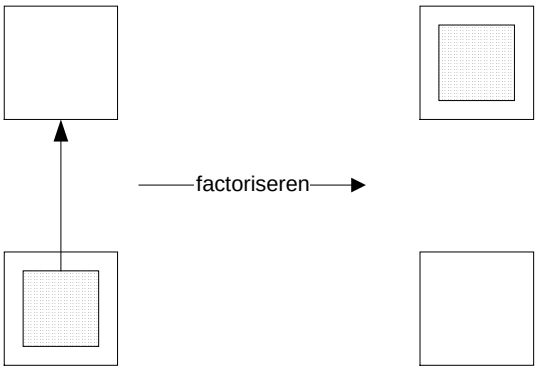
```
A = 2·B
B = 3
```

Dan een gefactoriseerde situatie (A gebruikt C en B gebruikt C):

```
A = 2·C
B = C
C = 3
```

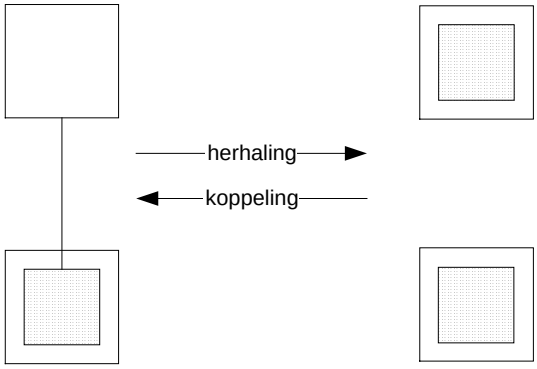
A is nu niet meer afhankelijk van B.

Een voor de hand liggende manier van factorisatie, die daarom misschien soms over het hoofd wordt gezien, is gewoon het verplaatsen van deelsystemen.



Figuur 54: factorisatie kan modulesamenstelling optimaliseren

Een belangrijke constatering is dat factorisatie niet gratis is. Het is altijd een afweging tussen herhaling en koppeling. Het vermeerderen van koppeling vermindert de herhaling en andersom. De complexiteit van de structuur verandert niet door te factoriseren.



Figuur 55: factorisatie kan herhaling ruilen voor koppeling

6.4 Parameterisatie

Naast dat het mogelijk is om bijvoorbeeld overeenkomsten buiten haakjes halen, te factoriseren, is het ook mogelijk om juist op de verschillen te concentreren. Zonder de verschillen blijven namelijk twee gelijke modules achter en die kunnen worden vervangen door een enkele '*multifunctionele*' module. Incomplete modules weliswaar, maar het ontbrekende deel kan worden gerepresenteerd door een parameter, een onbepaalde waarde. Een incomplete module is niet te gebruiken, het ontbrekende deel moet daarvoor eerst worden ingevuld. Pas na substitutie van een specifieke, bepaalde waarde als parameter wordt het een bruikbare module. In de wiskunde zou zoiets er als volgt uitzien:

```
f(x) → 2x + 3
```

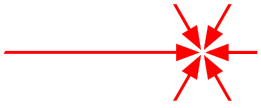
Dit is een wiskundige functie. De functie heeft geen waarde of uitkomst omdat de parameter onbepaald is, de functie kan dus niet als zodanig worden gebruikt in een berekening. Pas bij substitutie van een bepaalde parameter heeft de functie een uitkomst.

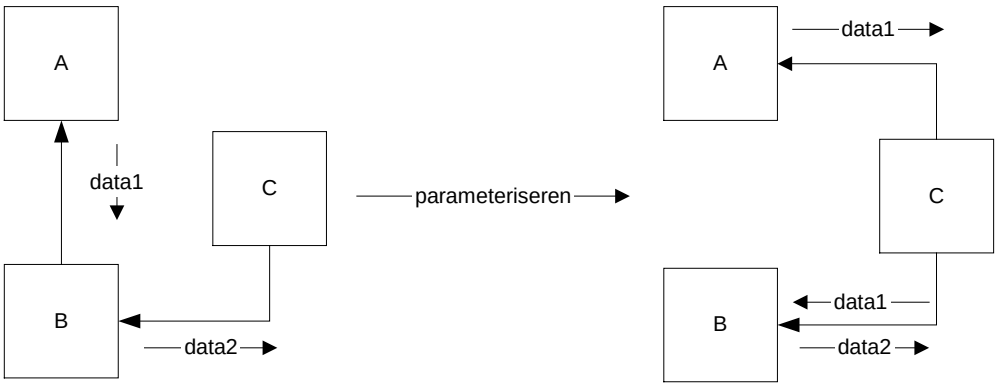
```
f(4) = 2.4 + 3 = 11
```

Dit heet *parameteriseren*. Parameteriseren werkt als volgt. Een module heeft soms voor de uitvoering van een opdracht andere modules nodig. Hij heeft dan gegevens (of materie, of energie) nodig en haalt die op via een koppeling. De module is dan afhankelijk van een andere module. De dingen die de module via een koppeling ophaalt, kunnen hem echter ook samen met de opdracht als invoer worden aangeboden in de vorm van een *parameter*.

In principe kan élke module op die manier onafhankelijk blijven van andere modules. De techniek kan dus worden gebruikt om modules contextvrij te maken, te ontkoppelen van hun omgeving. Deze techniek kan niet onbeperkt worden toegepast, uiteindelijk moet er natuurlijk ergens in het systeem een module zijn die de juiste gegevens ophaalt en dus wel aan zijn omgeving moet worden gekoppeld. De techniek, parameterisatie, is een basistechniek bij het maken van herbruikbare modules.

Koppeling kan worden vervangen door een parameter.

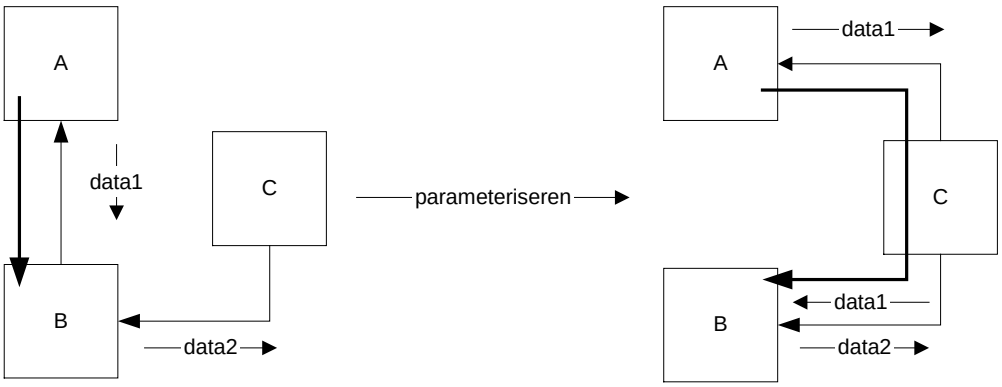




Figuur 56: door parameterisatie kunnen modules onafhankelijk worden gemaakt.

De bovenstaande figuur heeft uitleg nodig. In de oorspronkelijke situatie is C afhankelijk van B, en B afhankelijk van A. C vraagt bijvoorbeeld iets aan B (data2), en B heeft om te kunnen antwoorden informatie nodig van A (data1). B haalt dat zelf op. Als de koppeling van B van A echter ongewenst is, dan kan de vraag van C aan B worden geparameteriseerd. Als C nu een vraag stelt aan B, dan moet C daarvoor eerst informatie vragen aan A. Deze informatie geeft C dan bij het stellen van de vraag aan B door, als parameter van de vraag. B hoeft dan zelf geen informatie mee op te halen bij A. Dit lijkt goochelen, en dat is het ook. Er gebeurt uiteindelijk niets anders, alleen de koppelingen liggen anders en dat kan een voordeel zijn. B is nu niet meer afhankelijk van A. B is herbruikbaar zonder A.

Een kenmerkende eigenschap van parameterisatie is, net als bij factorisatie, dat de verschillende *stromen* vóór en ná factorisatie nog steeds in dezelfde modules beginnen en eindigen. Parameterisatie verlegt slechts de tussenpunten van een stroom, niet de begin- en eindpunten. Hierdoor kan de structuur na parameterisatie hetzelfde blijven werken maar met een andere verdeling van koppelingen. Vergelijk in onderstaande figuur de begin- en eindpunten van de gegevensstroom ‘data1’ die is aangegeven met een dikke pijl.



Figuur 57: na parameterisatie zijn de begin- en eindpunten nog steeds hetzelfde

Parameterisatie verlegt afhankelijkheden. Gegevens, energie of materie stromen via andere routes maar de routes hebben dezelfde begin en eindpunten.

Een voorbeeld in termen van wiskundige expressies, eerst de uitgangssituatie (C gebruikt B, B gebruikt A):

```
A       = 1
B       = 2·A
C       = 3·B
```

Dan een geparameteriseerde situatie (C gebruikt B én A, B heeft geen afhankelijkheden meer, maar is geparameteriseerd):

```
A       = 1
B(x)    = 2·x
C       = 3·B(A)
```

B is nu onafhankelijk van A en daardoor ook herbruikbaar in contexten waar A niet bestaat. Bijvoorbeeld de volgende:

```
D       = 4
E       = 3·B(D)
```

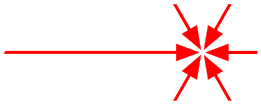
Parameterisatie is het vervangen van contextafhankelijkheid door een parameter, zodanig dat de geparameteriseerde modules contextonafhankelijk worden gemaakt zonder dat hun functie verandert.

6.5 Symmetrie en compressie

Intuïtief gezien is symmetrie een begrip dat vooral wiskundige spiegelbeelden beschrijft, maar in de formele zin is symmetrie een veel ruimer begrip. Naast wiskunde en geometrie zijn er nog veel meer vakgebieden te bedenken waar symmetrie of symmetrische bewerkingen kunnen worden herkend. Symmetrie refereert aan regelmaat in structuur in het algemeen.

Kenmerken van symmetriebewerkingen zijn dat het origineel ongeschonden blijft en dat er regelmatige patronen ontstaan door eenvoudige 'vermeerderende' bewerkingen op het origineel of uitgangspunt. De linkerhelft van een perfect gezicht, verkregen door de rechterhelft te spiegelen. Een taart van twaalf punten, opgebouwd door één punt twaalf maal een stukje rondom zijn punt te draaien. Of een kubus van duizend perfecte blokjes, in gedachten verkregen door eerst tien van die blokjes naast elkaar te leggen, daarna tien van die rijen op elkaar te stapelen en tot slot tien van die stapels achter elkaar te leggen. Duizend blokjes, gerepresenteerd door drie eenvoudige bewerkingen en één zo'n blokje. Behalve spiegeling, rotatie en translatie zijn er echter nog meer bewerkingen te bedenken die patronen doen ontstaan vanuit een origineel dat zelf bij de bewerking ongeschonden blijft.

Symmetrie ontstaat door vermeerderende bewerkingen op een eenvoudiger patroon. Onder symmetrie kan daarom worden verstaan alle vormen van *regelmaat* die zijn te vervangen door compactere constructies zonder symmetrie. Een uitgangspunt en een vermeerderende bewerking zijn voldoende om de symmetrie weer te kunnen herstellen.



Symmetrie is een patroon dat ontstaat door vermeerderende bewerkingen op een eenvoudiger patroon. Symmetrie is regelmaat.

Symmetrie kan zich, net als herhaling, diversiteit en koppeling, uitstrekken over alle dimensies van het systeem, ook in de dimensies tijd en schaal. Een voorbeeld van temporele symmetrie of symmetrie in tijd, is bijvoorbeeld periodiciteit. De beweging van een ideale slinger is bijvoorbeeld symmetrisch ten opzichte van een bepaald tijdstip en de beschrijving van zijn beweging is terug te brengen tot een enkele uitslag omdat elke uitslag hetzelfde zal zijn. Hetzelfde geldt voor de beweging van de secondewijzer van een klok, of de grote wijzer. Het is niet nodig om alle minuten te kennen die hij in zijn levensduur zal aanwijzen, slechts één minuut en een 'vermeerderende bewerking' kunnen volstaan. De stand van de wijzer bij alle andere minuten, nog te komen of reeds gepasseerd, kan daaruit worden herleid.

Herstructurering van structuur is gebaseerd op symmetrie. 'Herstructureerbare' structuren kunnen worden gevonden door te zoeken naar *symmetrie* of juist het aanbrengen van symmetrie. Symmetrie kan namelijk altijd compacter of anders worden voorgesteld. Namelijk zonder regelmaat, patronen en herhalingen. Hoe meer symmetrie er in een structuur voorkomt, hoe minder unieke deelstructuren er nodig zijn om samen met eenvoudige transformaties de originele structuur zonder symmetrie te kunnen representeren.

6.5.1 Compressie als basis van herstructurering

In een sterk symmetrische structuur kan vanuit een klein deel van de structuur door eenvoudige transformaties de gehele structuur worden gereconstrueerd. Andersom kan een sterk symmetrische structuur dus ook worden gereduceerd tot een veel compacter structuur met dezelfde 'informatiewaarde'. Deze manier van herformulering heet *comprimeren*. Compressie maakt gebruik van de symmetrie van een systeem door het eenvoudiger te representeren *zonder dat er informatie verloren gaat*. De oorspronkelijke structuur is altijd weer te reconstrueren.

Comprimeren is het herleiden van de symmetrische constructies in een systeem tot hun asymmetrische uitgangspunt zodat er geen symmetrie (herhaling of regelmaat) meer aanwezig is.

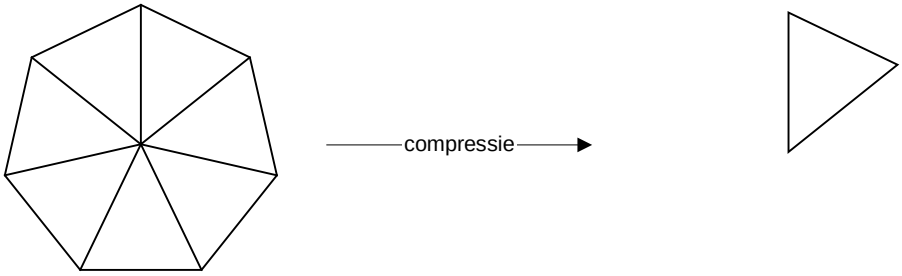
Compressie kan goed worden gecombineerd met abstractie. Door abstractie kan de effectiviteit van compressie worden verhoogd. Een voorbeeld. Met PKZIP kunnen tekstbestanden worden gecomprimeerd tot tien, twintig procent van hun oorspronkelijke grootte en zonder verlies weer worden gedecomprimeerd. Dit is mogelijk omdat een tekstbestand kennelijk veel 'symmetrie' of regelmaat bevat.

De toepassing van PKZIP en het bijbehorende ZIP-formaat op geluids- of beeldinformatie, dus bijvoorbeeld op een bitmap- of wave-bestand, levert echter veel slechtere resultaten. Geluid en beeld is kennelijk moeilijk te comprimeren. Er is dus weinig regelmaat in beeld- en geluidsinformatie te vinden. Om dit soort gegevens wel te kunnen comprimeren, is een aanvullende maatregel nodig. Hiervoor kan een MPG- of MP3-formaat worden gebruikt. Deze formaten reduceren beeld- en geluidsinformatie wél tot tien, twintig procent van hun oorspronkelijke grootte.

Het maken van een MPG-bestand van beelden (of een MP3-bestand van geluid) gebeurt echter door vóór compressie eerst abstractie toe te passen. Eerst worden alle miljoenen kleuren ‘geabstraheerd’ totdat alleen nog maar de essentiële kleuren overblijven. De miljoenen kleuren van het origineel worden afgerond naar slechts een paar kenmerkende kleuren en dan blijkt het beeld wel goed te kunnen worden gecomprimeerd. Niet onlogisch, want minder verschil betekent meer overeenkomst. Minder verschillende kleuren betekent dus meer gelijke kleuren, en dat is regelmaat, en die kan worden gecomprimeerd.

Om het mogelijk te maken om gecomprimeerde symmetrie weer te kunnen reconstrueren, is er een regel nodig in de betekenis van de notatie van de structuur. Een voorbeeld van zo’n regel is de middellijn of centerlijn die werktuigbouwkundig constructeurs gebruiken in hun bouwtekeningen. Deze lijn heeft geen betekenis in de getekende structuur maar hoort bij de tekenwijze en dient om bepaalde dingen gecomprimeerd weer te geven. Dit soort elementen heten daarom abstracte elementen. In principe hoeft de tekening slechts een helft van een symmetrisch figuur én zo’n middellijn te bevatten. De rest is duidelijk. Een ander voorbeeld is de herhalingsyntaxis in een computertaal (`for a:=1..10 do`, `while (a<10) do`). Deze modelleert herhaling (*temporele symmetrie*). Het is dankzij een regel in het model (de betreffende syntaxis) niet nodig om elke herhaling expliciet uit te schrijven.

Elke vorm van symmetrie waarvoor de notatie van de structuur een regel heeft, kan worden verwijderd uit de weergave en compacter door het asymmetrische uitgangspunt worden gerepresenteerd. Hoe meer van deze regels het model heeft, hoe minder symmetrie het hoeft te bevatten en hoe compacter het model kan zijn.



Figuur 58: compressie is het herleiden van symmetrie tot het asymmetrische uitgangspunt

Een duidelijke en eenvoudige vorm van compressie is de volgende wiskundige herformulering:

```
+ 2 + 2 + 2 + 2 + 2 = 5 · 2
```

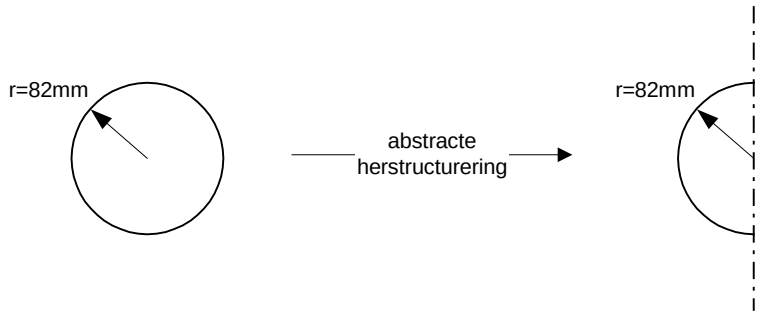
Dankzij een *compressieregel* in de notatie (de vermenigvuldiging) kan een herhaalde optelling zonder herhaling worden genoteerd. Dankzij de betekenis van de vermenigvuldiging is ook de omgekeerde transformatie (expansie) weer mogelijk:

```
5 · 2 = + 2 + 2 + 2 + 2 + 2
(patroon = + 2, herhaling = 5, regel = ·)
```

6.6 Abstracte en concrete herstructurering

Zoals een systeem kan worden geherstructureerd, zodanig dat de functie ervan niet verandert, zo kan ook een model worden veranderd, zonder dat het een ander systeem weergeeft. De meeste modellen moeten bijvoorbeeld regelmatig worden veranderd of aangepast. Om dat gemakkelijk te kunnen doen, moeten ze weinig herhaling (*redundantie*) bevatten. In dat geval is het wenselijk om het model zo compact mogelijk te noteren, echter zonder dat er een ander systeem wordt weergegeven!

Voor het veranderen van de structuur van een model kunnen dezelfde technieken worden gebruikt als voor het veranderen van de structuur van het systeem zelf. Het klinkt allemaal als iets zeer exotisch dat alleen is weggelegd voor alleen de allerbeste ontwerpers, maar het is gewoon een normale techniek die iedere ontwerper kan gebruiken. Zie het ongeveer als volgt: een bouwtekening kan mét en zonder spiegellijnen en gespiegelde elementen worden getekend, maar het geeft in beide gevallen nog steeds hetzelfde systeem weer. Dit is een beetje flauw voorbeeld, maar geeft toch duidelijk aan dat een model kan worden veranderd zonder dat het een ander systeem weergeeft.



Figuur 59: een eenvoudige vorm van abstracte herstructurering (twee weergaven van hetzelfde 'systeem')

Nu is het voorbeeld van de spiegellijn wellicht voor de hand liggend, maar later bij het ontwerpen van softwaresystemen zullen technieken worden gegeven die veel minder makkelijk worden herkend als ‘gewoon een herformulering van het model’. Een voorbeeld daarvan is overerving in een programmeertaal.

Wanneer een herstructurering zichtbaar is in de concrete structuur ('het systeem') dan wordt het *concrete herstructurering* genoemd. Wanneer de herstructurering is bedoeld om ‘slechts’ de notatie van een model te optimaliseren, en dus slechts zichtbaar is in de abstracte structuur ('een model'), dan wordt het *abstracte herstructurering* genoemd. In dat geval is de herstructurering onzichtbaar buiten het betreffende model.

> *Er bestaat abstracte en concrete herstructurering. Concrete herstructurering betreft het concrete systeem en is daarom zichtbaar in alle weergaven van het systeem. Abstracte herstructurering heeft zijn uitwerking slechts in het betreffende model.*

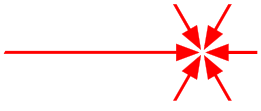
Een goed voorbeeld van het herformuleren van modellen zonder dat ze een ander systeem gaan weergeven, wordt gegeven door de menselijke *cognitie*. De menselijke

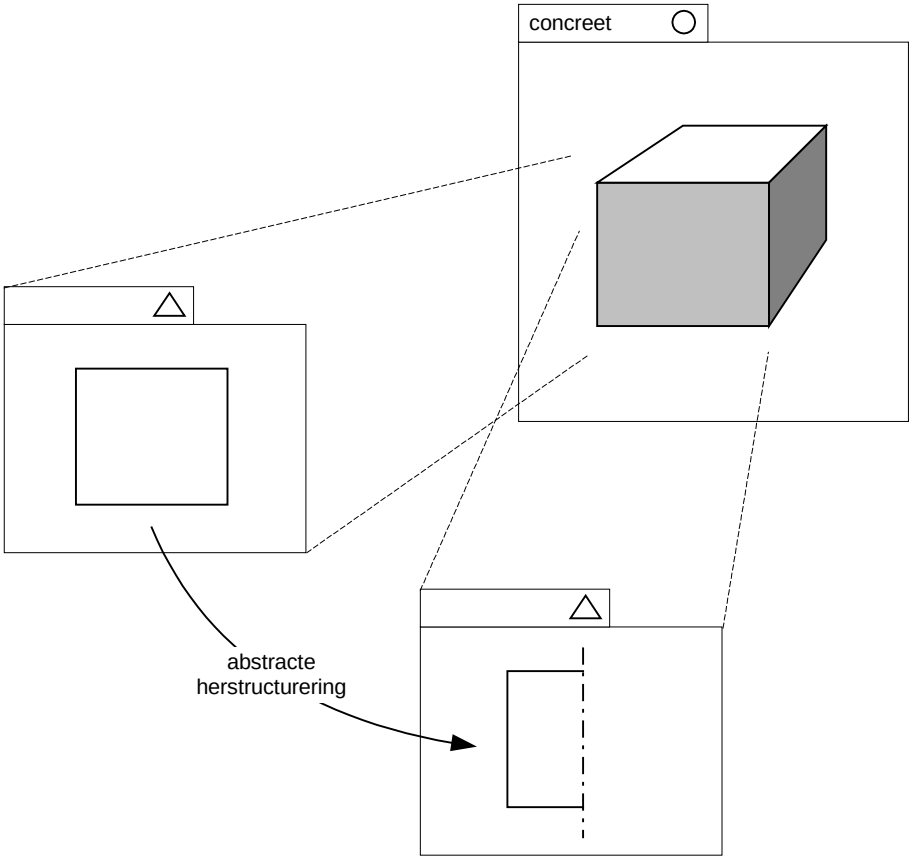
hersenen vormen *concepten* om dingen uit de werkelijkheid mee te representeren. Ze maken een typerend model van de werkelijkheid op basis van concepten. Concepten vormen een model dat het *Real Life System* (RLS) heet. Doordat de mens telkens met nieuwe aspecten van dezelfde werkelijkheid wordt geconfronteerd, moet dat model echter voortdurend worden aangepast. Het RLS wordt in principe steeds completer. Wijsheid komt met de jaren en misschien wel daarom. Om dit zo efficiënt mogelijk te kunnen doen, vormen de hersenen bij voldoende gelijkenis (*symmetrie*) tussen twee concepten automatisch een nieuw 'superconcept'. Dit superconcept is een generalisatie van de twee gelijkende concepten en bevat hun onderlinge overeenkomsten. De oorspronkelijke concepten worden gerelateerd aan dit superconcept, zodanig dat de eigenschappen van het *superconcept* worden ‘overerft’ door de afgeleide concepten. Een auto en een fiets zijn beide voertuigen. Zowel de auto als de fiets hebben de eigenschappen van een voertuig. De twee oorspronkelijke concepten worden echter ‘onthouden’ als drie concepten (auto, fiets, voertuig). Als er nu een wijziging moet worden aangebracht die zowel het concept auto als het concept fiets betreft, dan hoeft dit maar in één concept te worden gedaan, namelijk in voertuig. Dit generaliseren en specialiseren wat de hersenen doen, kan worden gezien als herformuleren van een model met abstracte factorisatie. Deze stap verandert niets aan de werkelijkheid, alleen aan het mentale model van de werkelijkheid dat de hersenen opstellen!

6.6.1 Structuurdomeinen

Concrete herstructurering heeft als doel het wijzigen van de structuur van het systeem maar kan in elk gewenst model van het systeem worden uitgevoerd. In elk gewenst structuurdomein. De ontwerper kan er voor kiezen om de concrete herstructurering uit te voeren in het abstracte of het typische domein en zolang zijn handelingen de juiste uitwerking hebben op de structuur van het uiteindelijke systeem, in het concrete domein, is het concrete herstructurering. Het uitvoeren van een concrete herstructurering in één bepaald model betekent wél altijd dat álle andere modellen moeten worden gecontroleerd op consistentie met de gewijzigde structuur van het systeem.

Abstracte herstructurering is meer een notatietruc die juist níet zichtbaar moet zijn in andere soorten modellen of domeinen, dan waar het in wordt uitgevoerd. Abstracte herstructurering kan dan ook slechts in het betreffende domein worden uitgevoerd en is onzichtbaar in het systeem of andere modellen.





Figuur 60: abstracte herstructurering is onzichtbaar in een andere weergaven: het beïnvloedt niet datgene wat wordt weergegeven door het model

6.6.2 Abstracte en concrete elementen

Voor het toepassen van abstracte herstructurering zijn altijd elementen nodig die niets met het systeem zelf te maken hebben. De spiegellijnen van een bouwtekening, het alfabet van een Huffmancompressie en ook de overervingnotatie van een objectgeoriënteerd model zijn zulke elementen. Dergelijk 'hulpelementen' heten *abstracte elementen*. Abstracte elementen zijn alleen zichtbaar in het betreffende model. Dit in tegenstelling tot de concrete elementen die ook in het concrete domein zichtbaar zijn.

> *Er bestaan abstracte en concrete elementen. Concrete elementen zijn (ook) zichtbaar in het concrete domein, abstracte elementen alleen in het betreffende model.*

Abstracte elementen ontstaan door een bepaalde symmetrische structuur zonder symmetrie te representeren (compressie). Abstracte elementen representeren de

vermeerderende bewerkingen die nodig zijn om de oorspronkelijke structuur weer te kunnen herstellen.

6.6.3 Abstracte factorisatie

De verschillende modules van een model kunnen specificaties met elkaar gemeen hebben. Het is mogelijk om deze overeenkomende specificaties als het ware 'buiten haakjes' te brengen. Specificaties kunnen op die manier worden ontbonden in factoren. Dit is een vorm van *factoriseren* en modellen die daardoor ontstaan, heten *gefactoriseerde modellen*.

De gevonden overeenkomsten tussen specificaties worden bij factorisatie gerepresenteerd door een nieuwe specificatie die de overeenkomstige eigenschappen van de deels gelijkwaardige specificaties in zich heeft. De gefactoriseerde specificaties, die zijn ontdaan van hun overeenkomende eigenschappen, erven daarbij als het ware de eigenschappen van deze nieuwe specificatie. Er is wel een speciale notatie nodig om uit het model te kunnen opmaken dat het *gefactoriseerde specificaties* betreft. In de wiskunde zijn dat de ronde haakjes. In de software-technologie is dat de *overervingnotatie*.

> *Abstracte factorisatie is het veranderen van de samenstelling van deelmodellen zonder dat het model een ander systeem gaat weergeven.*

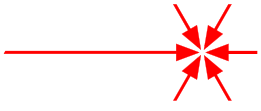
Factorisatie van modellen werkt twee kanten op. Behalve dat overeenkomsten kunnen worden verwijderd uit een bestaand model, kunnen overeenkomsten ook worden vermeden bij aanvullingen op het model. Bijvoorbeeld in een typerend model. Nieuwe typeringen kunnen gebruik maken van de eigenschappen van bestaande typeringen, indien er overeenkomsten zijn. In een gefactoriseerd model hoeven telkens alleen nog maar de verschillen te worden gespecificeerd!

Als een systeem wordt gebouwd vanuit een model, zoals het geval is bij het bouwen van een huis vanaf een bouwtekening, dan kan achteraf nooit worden gezegd of de notatie van het model gefactoriseerd was of niet. Zoals aan het huis niet meer is te zien of de bouwtekening spiegellijnen, middellijnen of andere abstracte elementen bevatte. De toepassing van abstracte factorisatie leidt dus niet direct tot betere of slechtere systemen. De keuze om een model te factoriseren wordt dus slechts bepaald door de wens om het model gemakkelijk te kunnen veranderen of niet. De prijs van factorisatie is vaak een complexer model (maar waarop bepaalde handelingen eenvoudiger kunnen worden uitgevoerd).

> *Aan een systeem is nooit te zien of het is gemaakt aan de hand van een abstract gefactoriseerd model of niet.*

6.6.4 Abstracte parameterisatie

Een systeem kan worden geparameteriseerd, dit heet concrete parameterisatie. Contextafhankelijke informatie wordt dan aangeboden bij de invoer die het systeem ontvangt (in plaats van dat het systeem de informatie zelf ophaalt). Hierdoor ontstaan contextonafhankelijke, herbruikbare deelsystemen: deelsystemen die in verschillende



systemen kunnen worden gebruikt. Een model kan ook worden geparameteriseerd. In dat geval ontstaan er herbruikbare deelmodellen: deelmodellen die in modellen van verschillende systemen kunnen worden gebruikt.

Bij concrete parameterisatie worden delen van het systeem voorzien van een parameter. De delen kunnen alleen worden gebruikt als andere delen van het systeem de parameter van een zinnige waarde voorzien. Een zelfde redenatie gaat op voor een geparameteriseerd modelelement, pas na invulling van een bepaalde parameter ontstaat een volwaardig modelelement dat kan worden gebruikt in een specifiek model. De beste manier om een geparameteriseerd modelelement te zien, is dan ook als een *sjabloon*.

> *Abstracte parameterisatie is het herbruikbaar maken van (deel-) modellen door bepaalde eigenschappen van het model onbepaald te laten en te representeren met een parameter.*

Van een bepaald systeem kan achteraf nooit worden gezegd of het model dat werd gebruikt om het systeem te bouwen abstract geparameteriseerde modelelementen bevatte of niet. De toepassing van abstracte parameterisatie leidt dus niet direct tot de constructie van betere of slechtere systemen. Wel leidt het hergebruik van modelelementen indirect tot betere systemen als het goed geteste en reeds bewezen elementen betreft.

De keuze om in een model gebruik te maken van geparameteriseerde modelelementen wordt voornamelijk bepaald door de wens om het model eenvoudiger veranderbaar te maken of niet. De prijs van abstracte parameterisatie is een complexer model.

> *Aan een systeem is nooit te zien of het is gemaakt aan de hand van een geparameteriseerd model of niet.*

6.7 Samenvatting

Dankzij het principe van algemeenheid van structuur is het mogelijk om algemeen geldige technieken te formuleren. Een belangrijke techniek is het herstructureren van structuur. Herstructureren is het aanpassen van een structuur zonder dat de functie van de structuur verandert.

Er zijn waarschijnlijk maar een paar fundamentele manieren om structuur te herstructureren waarvan er in dit boek twee worden beschreven. De onderstaande tabel toont de fundamentele herstructureringstechnieken:

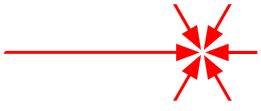
Herstructureringstechnieken		
Factoriseren	Hoe	Door de samenstelling van modules te veranderen.

Herstructureringstechnieken		
	Waarom	Het resultaat van factorisatie is een andere structuur met dezelfde werking. Factorisatie kan twee richtingen op werken: naar een structuur met minimale herhaling óf naar een structuur met minimale koppeling. Minimale herhaling betekent automatisch maximale koppeling, en minimale koppeling betekent automatisch maximale herhaling.
Parameteriseren	Hoe	Door contextafhankelijkheid te vervangen door een parameter.
	Waarom	Het resultaat van parameterisatie is herbruikbaarheid en vermindering van herhaling in een structuur. Een geparameteriseerde structuur is onafhankelijk van specifieke context of omringende structuur en kan dus (door invulling van de juiste parameter) in andere contexten ook worden gebruikt.

De notatie van een model heeft volgens het principe van algemeenheid van structuur zelf ook structuur. En alle bewerkingen die geldig zijn op de structuur van een systeem (een concrete weergave), zijn (in principe) ook geldig op de structuur van het model.

Het veranderen een model hoeft geen gevolgen te hebben voor de weergegeven structuur (het systeem). Op basis van dit onderscheid bestaan twee soorten herstructurering:

Herstructureringssoorten		
Abstract	Hoe	Abstracte herstructurering is het veranderen van de structuur van het model (een abstracte weergave) zonder dat dit zichtbaar / merkbaar is in andere weergaven. Abstracte herstructurering kan alleen maar worden uitgevoerd in de betreffende abstracte weergave.
	Waarom	Om het model de juiste structuur te geven. Weinig herhaling in het model als er veel veranderingen worden verwacht. Weinig koppelingen tussen deelmodellen als er herbruikbare deelmodellen nodig zijn.



Herstructureringssoorten		
Concreet	Hoe	Concrete herstructurering is het veranderen van de structuur van het concrete systeem. Concrete herstructurering kan in elke gewenste weergave worden uitgevoerd maar betekent wel dat alle andere weergaven moeten worden gecontroleerd op consistentie.
	Waarom	Om het systeem de juiste structuur te geven. De reden dat we modelleren!

Elementen van een model die niet zichtbaar zijn in het concrete domeinen zijn abstracte elementen. Elementen die onderdelen van het concrete systeem modelleren zijn concrete elementen.

Aan de basis van deze technieken en alle andere herstructureringstechnieken ligt symmetrie en de techniek om die anders te kunnen noteren: compressie en decompressie (expansie).

7. Kwaliteit van structuur

'You know you have achieved perfection in design, not when you have nothing more to add, but when you have nothing more to take away...'

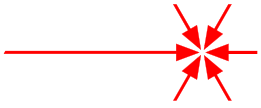
Antoine de Saint Exupery

Hoofdstukwijzer

7.1 Eenvoud

De belangrijkste eigenschap van elk ontwerp is *eenvoud*. Eenvoud werkt preventief tegen elke fout en vergissing die we maar kunnen bedenken.

Eenvoud is het kenmerk van het ware.



En daaruit volgt dan ook meteen het beste advies voor de ontwerper.

> *In de beheersing toont zich de meester.*

7.2 Fysieke en virtuele systemen

Systemen kunnen op verschillende manieren worden geclassificeerd, maar een bijzondere opdeling vanuit het gezichtspunt van kwaliteit van softwaresystemen is de volgende:

- Fysieke systemen

  Een *fysiek systeem* neemt ruimte in en is tastbaar. Een fysiek systeem heeft een bepaalde ruimtelijke afmeting (*geometrie*). Voorbeelden van fysieke systemen zijn de dingen om ons heen die we kunnen aanraken.

- Virtuele systemen

  Een *virtueel systeem* neemt geen ruimte in maar bestaat slechts uit gegevens. Een virtueel systeem heeft geen ruimtelijke afmetingen. Voorbeelden van virtuele systemen zijn softwaresystemen, rekenmodellen en theorieën.

Een goed besef van de zojuist genoemde verschillen tussen deze twee soorten systemen is heel belangrijk. Het vergroot de controle die de ontwerper heeft over de kwaliteit van het ontwerp. Een softwaresysteem is een virtueel systeem en heeft net als elk ander virtueel systeem een onbeperkte graad van structurele vrijheid. Dat maakt het moeilijker te ontwerpen dan fysieke systemen. De grootste beperking en daardoor een belangrijke invloed bij de constructie van een fysiek systeem is namelijk de *afmeting* van onderdelen. Door de fysieke afmetingen van onderdelen is het moeilijk om elke gewenste koppeling te realiseren. Ontkoppeling (modulariteit) worden in fysieke systemen voor een groot deel automatisch afgedwongen door fysieke beperkingen. Het is in fysieke systemen nou eenmaal makkelijker om naastliggende delen te verbinden dan niet-naastliggende. Virtuele systemen hebben geen afmetingen en dus ook geen fysieke beperkingen. Elke koppeling is even gemakkelijk te maken! Een bepaalde regel van een wetgeving kan even gemakkelijk naar de vorige regel verwijzen als naar de allerlaatste, vier wetboeken verderop. (En in wetboeken en andere ambtelijke regelgeving doen ze dat ook. Ambtenaren hebben een hekel aan eenvoud.) Een procedure in een softwareprogramma kan zonder problemen worden gekoppeld aan willekeurig welke andere procedure. Er zijn geen fysieke beperkingen. Al snel ontstaat in virtuele systemen een Spaghettistructuur.

> *Een virtueel systeem is geometrieloos en heeft daardoor een onbeperkte graad van structurele vrijheid: het systeem legt geen fysieke beperkingen op aan structuur.*

De complexiteit van virtuele systemen is in het algemeen veel groter dan die van fysieke systemen en het ontbreken van fysieke afmetingen is een belangrijke reden daarvoor.

> *Alleen beheersing van de ontwerper kan een ontwerp van een virtueel systeem (zoals software) eenvoudig houden.*

Een ander belangrijk verschil tussen fysieke en virtuele systemen is dat het fysieke systeem altijd als belangrijkste gezichtspunt de zichtbare opbouw heeft. Dit is voor iedere observeerder redelijk gelijk. Een virtueel systeem heeft geen 'normaal gezichtspunt', geen vanzelfsprekende manier om het systeem te bekijken. Het begrip en de beeldvorming van een virtueel systeem zullen daarom altijd afhangen van het gekozen gezichtspunt en de kwaliteit van de gekozen visualisatie.

7.2.1 Virtuele systemen en ruimtelijke verhoudingen

Virtuele systemen hebben geen ruimtelijke afmetingen (*geometrie*) of ruimtelijke verhoudingen (*topologie*). Deze eigenschap lijkt aantrekkelijk omdat er dan ook geen ruimtelijke eigenschappen zijn die bepaalde koppelingen minder aantrekkelijk (kostbaar) of onmogelijk maken. Elke gewenste verbinding is namelijk even gemakkelijk te maken zonder dat ruimtelijke beperkingen dit verhinderen. In de praktijk zorgt deze eigenschap echter voor een grotere complexiteit en is deze eigenschap van virtuele systemen ongewenst.

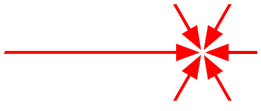
Virtuele systemen kunnen worden ontdaan van hun onbeperkte structurele vrijheid door ze schijnbare ruimtelijke verhoudingen toe te kennen. Door net te doen alsof een virtueel systeem wel afmetingen heeft, kunnen er eigenschappen zoals relatieve afstand, ‘boven’ en ‘onder’ worden onderscheiden. Wat een ontwerper in feite zou moeten doen, is het aanbrengen van schijnbare ruimtelijke verhoudingen. Als er duidelijke ruimtelijke verhoudingen in het systeem aanwezig zijn, is het gemakkelijker om koppelingen te beheersen. Dankzij de ruimtelijke verhoudingen is het namelijk mogelijk om onderscheid te kunnen maken tussen *lokale* en *interlokale koppelingen*.

Lokale koppelingen verbinden alleen aangrenzende of naastliggende modules terwijl niet-lokale koppelingen juist niet aangrenzende of naastliggende modules verbinden. Lokale koppeling is veel onschadelijker voor een systeem dan interlokale koppeling. In een systeem met slechts lokale koppeling is het veel gemakkelijker om de doorwerking van bepaalde effecten te volgen. Uitvergroten van het deel van het systeem met de te observeren module kan gebeuren zonder dat andere belangrijke modules uit beeld verdwijnen, deze liggen immers vlak naast de betreffende module.

> *Door te doen of een virtueel systeem ruimtelijke verhoudingen heeft, kan de eenvoud van het ontwerp beter worden afgedwongen.*

7.3 Kwaliteit van vorm

Een systeem met een bepaalde functie kan vele vormen hebben, afhankelijk van het gebruikte palet en de gekozen functionele decompositie die aan de vorm ten grondslag liggen, maar één ding is altijd belangrijk: de vorm moet evenredig zijn aan de functie. Een deelfunctie wordt dan verzorgd door een deelvorm, en een functieaspect door een vormaspect. Een veel gemaakte fout is dat functieaspecten in vormdelen worden ondergebracht of andersom. De klassieke fout is bijvoorbeeld alle soorten besturing van een systeem in een deelsysteem onder te brengen terwijl de soorten besturing functioneel gezien door het hele systeem verspreid liggen (en dus een aspect zijn in plaats van een deel).



De belangrijkste eigenschap van een vorm of een model van vorm is dus:

- Functionele evenredigheid

  Een goede vorm is evenredig met de functie van het systeem. Er is sprake van *functionele evenredigheid* als er in alle weergaven van de vorm (dus ook de abstracte en typerende modellen) een vaste en duidelijke verhouding is tussen de vormstructuur en de functionele structuur van het systeem.

7.3.1 Principe van evenredigheid van vorm en functie

Evenredigheid op zich is altijd een gewenste eigenschap, in elk aspect van het systeem en in elke fase van het ontwikkelingsproces. Alle structuren die betrokken zijn bij de ontwikkeling van een systeem moeten een goede *evenredigheid* vertonen. Van directory-structuren tot projectstructuren, van documentatiestructuur tot ontwerpstructuur en helemaal van functionele structuur tot vormstructuur.

> *Een vorm of weergave van vorm is van goede kwaliteit als de structuur ervan evenredig is met de functionele structuur van het systeem. Dit is het principe van functionele evenredigheid van vorm en functie.*

7.3.2 Functionele evenredigheid met het supersysteem

Een functiemodel is een model dat een structuur van functies weergeeft. In principe is elk functiemodel goed zolang het logisch is, een goede modulariteit heeft én niet strijdig is met de gewenste mogelijkheden of bestaande structuren en processen. Een goed functiemodel kan zo goed mogelijk voldoen aan alle eisen die aan het systeem worden gesteld. Het woord ‘alle’ is in dit geval het probleem. De eisen zijn vaak zo uitgebreid en divers, en soms zelf tegenstrijdig, dat het moeilijk is om een functiemodel te bedenken dat aan alle eisen evenveel recht doet. De echte waarde van het gekozen functiemodel zal dan toch pas na langere tijd van gebruik duidelijk worden. Om de juiste inschattingen te kunnen maken is ervaring en een goede kennis van het probleem nodig.

Het functiemodel bepaalt de concrete opbouw van het systeem. Functies die in het functiemodel als deelfuncties zijn (h)erkend, zullen ook in het concrete systeem als losse eenheden voorkomen (dat wil zeggen: als het een goed gebouwd systeem is). Een goed functiemodel bestaat dan ook uit deelfuncties waarvan de concrete deelsystemen die de betreffende functie leveren later individueel kunnen worden behandeld als ‘plug and play’ elementen, bijvoorbeeld in het geval van vervanging, aanpassing of hergebruik van zo’n deelsysteem. Daarvoor worden de functies zo gekozen dat ze zo min mogelijk koppeling en materie- data- of energieuitwisseling hebben met andere functies.

> *De aanpasbaarheid en herbruikbaarheid van (de delen van) het concrete systeem worden bepaald door het gekozen functiemodel van het systeem!*

De betere functiemodellen zijn in principe gebaseerd op de functionele structuur van de omgeving (het *supersysteem*) dat zij moeten ondersteunen. Een goede integratie van

een systeem met zijn omgeving zorgt ervoor dat het systeem in alle opzichte evenredig zal zijn met die omgeving: het zal gevoelig zijn voor dezelfde veranderingen of gebeurtenissen en het zal worden beschouwd als een voorspelbaar en handelbaar systeem door de mensen of systemen in zijn omgeving.

> *Een functiemodel wordt bij voorkeur gebaseerd op de structuur van het systeem dat het moet ondersteunen of de structuur van de omgeving, dat wil zeggen: van het supersysteem*

7.4 Functionele kwaliteit van systemen

Een systeem dat wordt gebouwd naar gebruikerspecificaties moet aan zeer belangrijke voorwaarden voldoen die het systeem relateren aan de specificatie:

- Correctheid

  Een *correct systeem* kan zijn functie volledig en naar wens vervullen. Correcte systemen werken precies zoals is gespecificeerd, zijn dus consistent met de specificatie.

  Correctheid mag nooit worden opgeofferd in het voordeel van andere eisen, maar een correct systeem is nog niet een goed systeem!

  Correctheid bestaat alleen als er een specificatie beschikbaar is waaraan deze te toetsen is!

- Robuustheid

  Een *robuust systeem* zal zijn functie kunnen blijven vervullen ook als er onvoorziene dingen gebeuren.

  Robuustheid is de bestendigheid tegen abnormale condities.

Robuustheid en correctheid zijn aanvullende eigenschappen. Correctheid betreft de werking ‘binnen’ specificatie en robuustheid de werking ‘buiten’ specificatie.

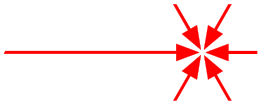
> *Een goed systeem is correct en robuust. Het werkt volgens specificatie en is bestand tegen ongespecificeerde gebeurtenissen.*

Een belangrijke opmerking bij correctheid is dat het grootste deel van de specificatie toch altijd impliciet zal (moeten) zijn. Een voorbeeld van impliciete specificaties zijn natuurkundige wetten of wiskundige regels en definities. Alles specificeren is onmogelijk en ongewenst.

7.5 Kwaliteit van structuur

Bij het ontwerpen van een systeem is het mogelijk om bepaalde kwaliteiten in de structuur ervan in te bouwen die het omgaan met het systeem vereenvoudigen. De belangrijkste eigenschappen zijn in dat opzicht: *hiërarchie* en *modulariteit*

In elk systeem is structuur te herkennen. Nog zonder naar de inhoud of werking van de structuur te kijken kan er al een kwaliteitsoordeel aan worden gegeven. Goede



structuur is eenvoudig of kan eenvoudig worden genoteerd. Elke structuur moet zodanig kunnen worden genoteerd dat er niet teveel informatie zichtbaar is en zonder dat er veel informatie ontbreekt in de weergave ervan.

7.5.1 Hiërarchische kwaliteit

Voor een goede begrijpelijkheid moet een structuur meerdere, duidelijk herkenbare niveaus van aggregatie hebben. Dat zorgt ervoor dat er nooit een té complexe structuur zichtbaar is; de ontwerper kan in zo'n geval terugvallen op een hoger niveau van aggregatie. Tegelijkertijd kan ook worden gekeken naar de details, de ontwerper kan dan juist afdalen naar een lager niveau van aggregatie.

Een structuur moet voldoende, duidelijk herkenbare niveaus van aggregatie hebben.

De *hiërarchische kwaliteit* van een structuur kan worden getoetst aan de hand van de volgende kenmerken:

- Begrijpelijk

  Een structuur is *hiërarchisch begrijpelijk* als er geen niveaus van aggregatie in voorkomen met een te hoge complexiteit. Er moeten dus voldoende niveaus van aggregatie te onderkennen zijn. Het kenmerk van een goede hiërarchie is het ontbreken van delen met complexe structuur.

- Deelbaar

  Een structuur is *hiërarchisch deelbaar* als er gemakkelijk verschillende niveaus van aggregatie in zijn te herkennen.

7.5.2 Modulaire kwaliteit

De *modulaire kwaliteit* van een systeem (de structurele kwaliteit op een gegeven niveau van aggregatie) kan worden getoetst aan de hand van de volgende kenmerken:

- Begrijpelijk

  Een systeem is *modulair begrijpelijk* als elke module los van de andere modules, in isolatie, kan worden begrepen en de structuur als geheel nog kan worden overzien.

  Dit is het basiscriterium waaraan elke module moet voldoen. Alle andere criteria zijn afgeleiden van dit criterium.

- Deelbaar

  Een systeem is *modulair deelbaar* als het gemakkelijk kan worden opgesplitst in kleinere, eenvoudigere en onderling losgekoppelde deelsystemen die redelijk zelfstandig kunnen worden behandeld. De modules moeten hiervoor zelfstandig en herkenbaar zijn.

- Samenstelbaar

Een systeem is *modulair samenstelbaar* als de modules van het systeem met de modules van andere systemen kunnen worden gecombineerd tot een geheel nieuw systeem.

- Coherent

  Een systeem is *modulair coherent* als de inhoud van modules bestaansafhankelijk is van de functie van de respectievelijke modules. Een coherente module bestaat uit alleen maar bestaansafhankelijke delen.

- Ontkoppeld

  Een systeem is *modulair ontkoppeld* als de modules zo min mogelijk andere modules nodig hebben om te kunnen functioneren en als er tijdens de communicatie zo min mogelijk informatie wordt uitgewisseld.

Koppelingen moeten in het systeem zoveel mogelijk lokaal van karakter zijn. Modules worden bij voorkeur zo min mogelijk gekoppeld aan andere en dan ook nog bij voorkeur alleen aan naastliggende modules. Het vaststellen van de reikwijdte van een koppeling is alleen mogelijk in een systeem waar één of andere topologie is vastgesteld.

In een modulaire structuur komen ook weinig tot geen *afhankelijkheidslussen* voor. Afhankelijkheidslussen zijn schadelijk voor de eenvoud en de veranderbaarheid. Ze kunnen ook leiden tot een oscillatie van effecten of een onvoorziene recursie van effecten. Structuren met afhankelijkheidslussen zijn veel moeilijker te begrijpen, te veranderen en te testen.

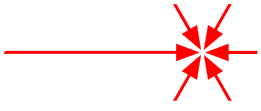
Modulariteit is een eigenschap die geldt voor een bepaald niveau van aggregatie. Elk niveau van aggregatie op zich moet een goede modulariteit hebben.

> *Elk niveau van aggregatie in het systeem moet een goede modulariteit hebben waarbij elke module op zichzelf staand, los van zijn specifieke of actuele context, moet kunnen worden begrepen.*

7.5.3 Balanceren van complexiteit

Door te herstructureren kunnen structuren worden veranderd zonder verandering van de functie van de structuur als geheel. Herstructurering is daarmee een krachtig hulpmiddel bij het maken van kwalitatief hoogwaardige systemen. Door te herstructureren kunnen bijvoorbeeld structuren die veel herhaling bevatten worden geformuleerd zonder herhaling. Dat maakt deze structuren beter te veranderen zonder *consistentieproblemen*. Een verandering hoeft immers maar op één plek in de structuur te worden aangebracht zodat het gevaar op inconsistente kopieën klein is.

Aan de andere kant is de prijs voor het verwijderen van herhaling, dat er meer koppelingen moeten worden aangebracht en dat maakt een structuur moeilijker te hergebruiken als deelstructuur in nieuwe structuren. Er komen minder autonome modules voor in een herhalingvrije structuur. Het maakt ook dat gebeurtenissen en dus ook ongewenste gebeurtenissen (fouten) door de structuur kunnen propageren en daarmee een veel groter deel van de structuur kunnen aantasten dan in een los



gekoppelde structuur. *Robuustheid* is dus moeilijker te garanderen in een structuur die is gefactoriseerd naar minimale herhaling.

Een belangrijke overweging bij het toepassen van herstructurering is de balans tussen al deze factoren. Wat is beter voor het gegeven systeem? Veel herhaling of juist veel koppelingen? Meestal zal het optimum tussen deze twee uitersten liggen. In het algemeen spelen bij het kiezen van de juiste vorm of structuur de volgende kwaliteitsafwegingen:

Minimale herhaling / maximale koppeling		Minimale koppeling / Maximale herhaling
Consistent	↔	Robuust
Compact / gekoppeld		Autonoom / ontkoppeld

In veel gevallen is factorisatie dus een afweging tussen goede modulariteit (autonome modules, losse structuur) en goede (impliciete) consistentie (weinig herhaling, sterk gekoppelde modules).

> *Een structuur kan worden gefactoriseerd naar minimale herhaling óf naar minimale koppeling, maar niet naar beide. Beide situaties hebben voor- en nadelen zodat er een zorgvuldig afweging nodig is om de juiste balans te vinden.*

Nu kan het natuurlijk altijd zo zijn dat een structuur onnodig veel herhaling of onnodig veel koppeling heeft. Er bestaan namelijk ook gewoon slechte structuren. Zeker in virtuele structuren zoals softwaresystemen loert het gevaar van een wildgroei van koppelingen, omdat er nu eenmaal geen fysieke beperkingen zijn die een willekeurige koppeling verhinderen. Er kan dan hier en daar verbetering worden aangebracht, maar als de structuur echt slecht is, moet er een nieuw ontwerp worden gemaakt. Of moeten er misschien andere bouwstenen worden gebruikt.

Om ervoor te zorgen dat er uiteindelijk een goede structuur ontstaat, kan een willekeurige (op gevoel en intuïtie gevormde) structuur soms eerst worden gefactoriseerd naar minimale herhaling. Daarvoor bestaan eenvoudige methoden, in de databasetechnologie wordt gewerkt met normaalvormen en deze kunnen met wat handigheid worden gebruikt voor de factorisatie van elke soort structuur, ook fysieke structuren. De resulterende structuur kan dan dienen als uitgangspunt voor de uiteindelijke factorisatieslag die herhaling en koppeling zorgvuldig balanceert.

> *Een veilig uitgangspunt van een goede structuur is een structuur die is gefactoriseerd naar minimale herhaling.*

7.6 Secundaire functionaliteit

Secundaire functionaliteit betreft de eigenschappen die niet de directe of primaire werking van het systeem betreffen. De factoren herbruikbaarheid en onderhoudbaarheid worden vaak genoemd als meest belangrijke.

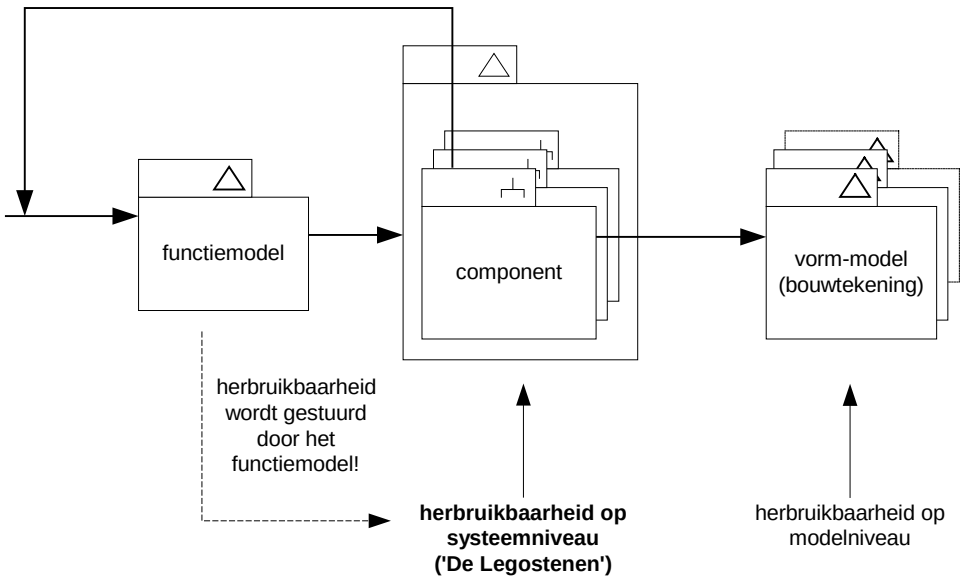
7.6.1 Herbruikbaarheid

Een belangrijke eigenschap van systemen en modellen is herbruikbaarheid. Herbruikbare systemen kunnen eenvoudig worden gebruikt om er nieuwe systemen, met uitgebreidere functionaliteit mee te maken. Herbruikbare modellen kunnen eenvoudig worden gebruikt om er nieuwe systemen, met uitgebreidere functionaliteit mee te modelleren.

Als een systeem herbruikbaar wordt genoemd, wordt bedoeld dat identieke kopieën of instanties van hetzelfde type van het systeem kunnen worden gebruikt om nieuwe systemen mee samen te stellen.

Herbruikbaarheid van systemen is iets anders dan van modellen. Herbruikbaarheid op *systeemniveau* is effectiever toe te passen dan die op *modelniveau*. Een doe-het-zelver die de bouwmarkt binnenloopt is typisch op zoek naar herbruikbare systemen (onderdelen) net als een elektronicus die bladert door een gids met standaard IC's. Een programmeur die de reference-manual van zijn programmeerbibliotheek doorbladert, is op zoek naar herbruikbare modellen (bijvoorbeeld handige classes of subroutines). Beide soorten herbruikbare elementen kunnen naast elkaar bestaan in hetzelfde ontwikkeltraject maar de meest effectieve en dus gewenste vorm van hergebruik is *assemblage* van nieuwe systemen uit componenten ofwel andere systemen. Om nieuwe systemen te maken op basis van bestaande deelsystemen is geen kennis nodig van de interen werking van die systemen. Om nieuwe modellen te maken op basis van bestaande modellen is wel zulke kennis nodig!

Herbruikbaar wil zeggen dat er eenvoudig nieuwe systemen of modellen mee kan worden gemaakt. Herbruikbaarheid bestaat op systeemniveau en op modelniveau.



Figuur 61: herbruikbaarheid is het meest waardevol op systeemniveau (componenten)

Succesvol hergebruik begint met het opstellen van slim functiemodel. Componenten zijn dragers van functies en worden gevonden in het functiemodel. In deel twee van dit boek zal dat uitvoerig worden besproken.

Een slim functiemodel is de sleutel tot effectief hergebruik.

Herbruikbaarheid gaat hand-in-hand met standaardisatie. Zonder gestandaardiseerde koppelingen, pasvormen en dergelijke, is herbruikbaarheid zinloos. Het complete stelsel van standaardisaties, regels en aannames dat daartoe dient, heet de technische architectuur van een systeem.

7.6.2 Onderhoudbaarheid

Een onderhoudbaar systeem is een systeem waarin gemakkelijk aanpassingen kunnen worden gedaan. Zowel wijzigingen van functie of vorm, als toevoeging als ook verbetering.

Wederom moet er duidelijk onderscheid worden gemaakt tussen de onderhoudbaarheid van een *systeem* en dat van de betrokken *modellen*. In de wereld van de fysieke systemen (elektronica, werktuigbouw) is onderhoudbaarheid van een systeem van veel groter belang dan de onderhoudbaarheid van de modellen (de bouwtekeningen). In de wereld van de virtuele systemen (software) is juist de onderhoudbaarheid van de modellen belangrijker, omdat de systemen vaak automatisch worden gegenereerd, vanuit de modellen en de systemen typisch niet toegankelijk zijn voor modificatie. Er zijn maar weinig systeembeheerders die de bits en bytes van een draaiend computerprogramma (een softwaresysteem) zullen veranderen. Eerder zal een programmeur opdracht krijgen de broncode (een model) van het systeem te veranderen en een nieuw systeem te genereren.

Vaak wordt gekozen voor herhalingvrije modellen. In zulke modellen komt geen herhaling van elementen voor. De symmetrie wordt uit een bouwtekening van een apparaat verwijderd door toevoeging van spiegellijnen. De herhaling van code van een softwaresysteem wordt verwijderd door toepassing van subroutines en overerving. Het idee hierachter is dat als er een wijziging nodig is, die wijziging maar op een enkele plaats in het model hoeft worden aangebracht. Deze aanname is niet geheel correct. Elke zinnige uitspraak hierover valt of staat met een juiste inschatting van de aard van de te verwachten wijzigingen. Er zijn best wijzigingen te verwachten waarvoor eerder verwijderde herhaling weer moet worden aangebracht.

Het beste wat een ontwerper kan doen als hij zo'n inschatting niet kan maken, is het systeem (en daarmee ook de modellen ervan) functioneel evenredig te maken, inclusief de natuurlijk aanwezige herhaling. Wijziging worden het systeem vaak opgelegd door zijn omgeving en door het systeem evenredig te maken met zijn omgeving, heeft het dezelfde gevoeligheid voor dezelfde soort wijzigingen. Wijzigingen die in de omgeving (het supersysteem) moeilijk zijn door te voeren, zijn dat ook in het systeem en andersom.

7.7 Kwaliteit van modellen

Er zijn goede modellen en er zijn slechte modellen. Een goed model is vooral duidelijk en subjectief eenvoudig. Door een model zo evenredig mogelijk te maken met het gemodelleerde systeem, is de herkenbaarheid optimaal.

- Gelijkvormigheid

  Als de vorm en opbouw van het model lijkt op dat van het systeem, is er sprake van *gelijkvormigheid* van het model. Gelijkvormigheid maakt een model gemakkelijker te begrijpen en ook gemakkelijker te relateren aan het systeem. In een stelsel van differentiaalvergelijkingen zal niet snel een model van een luidspreker kunnen worden herkend. In een netwerkmodel veel sneller. En in een schaalmodel nog veel sneller!

Een model is zoveel mogelijk gelijkvormig met het systeem dat wordt weergegeven. Dit is het principe van gelijkvormigheid van modellen.