Konstruktief ontwerpen met behulp van computerprogrammatuur

Hofmeyer, H.

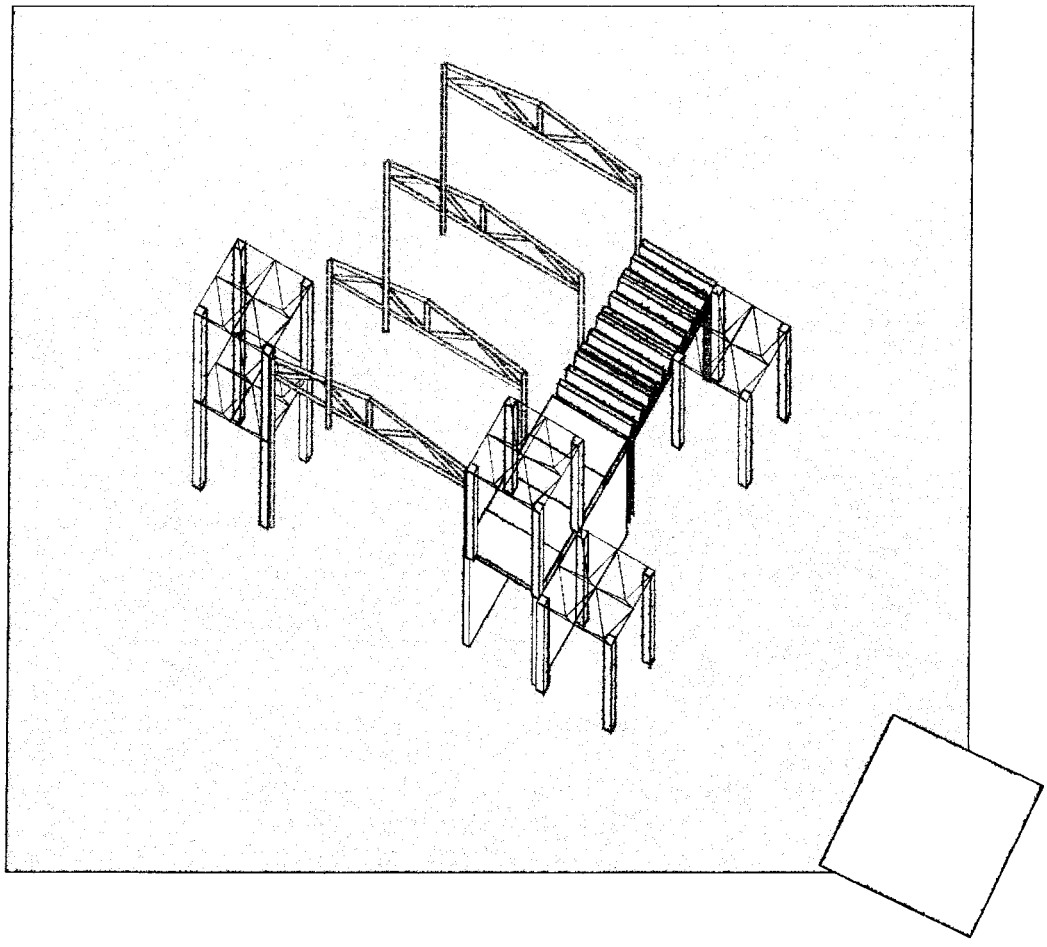
Award date:
1994

Faculteit Bouwkunde
Vakgroep
Konstruktief
Ontwerpen

Technische Universiteit
Eindhoven

Konstruktief ontwerpen met behulp van computerprogrammatuur

Verslag afstudeerproject



Herm Hofmeyer
332066
25 augustus 1994

Faculteit Bouwkunde
Vakgroep
Konstruktief
Ontwerpen

Technische Universiteit
Eindhoven

Konstruktief ontwerpen met behulp van computerprogrammatuur

Verslag afstudeerproject

Begeleidingscommissie:

Prof.dr.ir. H.S. Rutten
Prof.ir. A.J.M. Beulens
Ir. P.H. van Merendonk
Ir. H.J. Fijneman

Samenvatting

Doel

Antwoord geven op de vraag of er mogelijkheden zijn om de computer te gebruiken als hulpmiddel bij het konstruktief ontwerpen en of deze mogelijkheden te combineren zijn met space-allocation.

Methoden/middelen om doel te bereiken

Een mogelijkheid om de computer te gebruiken bij het konstruktief ontwerpen wordt ten dele uitgewerkt.

Na onderzoek naar bestaande konstruktieve ontwerpen en literatuurstudie, wordt de door het onderzoek verkregen kennis omgezet in processen en data. Mogelijke algoritmen worden ontwikkeld om het konstruktief ontwerpen te ondersteunen.

Er wordt een programma geschreven met de volgende functionele eisen:

- Op basis van een beperkte invoer van een aantal ruimten en de afmetingen hiervan worden een aantal konstruktief ontwerpen gepresenteerd, die globaal gedimensioneerd worden.

- De gebruiker moet tijdens het genereren van deze konstruktief ontwerpen kunnen ingrijpen en veranderingen mogelijk maken.

Dit programma bestrijkt het gehele proces konstruktief ontwerpen, maar bereikt bij elk aspect van het proces slechts een geringe diepgang:

- De omzet van de invoer (ruimten, afmetingen en voorwaarden) tot een ruimtelijk ontwerp.

- De konstruktie van het ruimtelijk ontwerp en globale dimensionering.

- De effecten van ruimtelijk ontwerp en konstruktief ontwerp op elkaar, en de terugkoppeling naar vorige ontwerpstappen.

- Evaluatie bij elke ontwerpstap en aanpassing ontwerp.

In het gehele afstudeerproject is het volgende van belang:

- De gevolgde werkwijze om kennis te verwerven en deze om te zetten in een programma is algemeen aanvaard en redelijk bekend.

- De manier waarop het geschreven programma en de algoritmen "konstrueren", de gekozen manier om een invoer van ruimten om te zetten in een konstruktief ontwerp, is slechts een exemplarisch resultaat. Er wordt duidelijk gemaakt dat er meer met een computer kan worden gedaan bij het konstruktief ontwerpen dan rekenwerk.

Resultaten/conclusies

De computer is te gebruiken als hulpmiddel bij het konstruktief ontwerpen en de gevonden methode is te combineren met space-allocation. Verder onderzoek zal vruchtbaar zijn.

Voorwoord

De beschrijving van dit afstudeerproject bestaat uit 3 delen:

Verslag afstudeerproject
Bijlage Gebruiksaanwijzing, code en toelichting bij programma*
Bijlage T8-T9-verslagen

* met programma's op diskette

De volgende personen van de begeleidingscommissie wil ik bedanken voor hun medewerking aan dit afstudeerproject:

Prof. dr. ir. H.S. Rutten zorgde voor een correcte algemene opzet en een gedetailleerde controle van het verslag.

Prof. ir. A.J.M. Beulens van de universiteit te Wageningen hielp bij het opzetten van een duidelijkere beschrijving van het programma.

Ir. P.H. van Merendonk van het gelijknamige adviesbureau te Amersfoort hielp bij de opzet van en corrigeerde de data- en procesmodellen. Verder gaf hij nauwgezet commentaar op het verslag.

Ir. H.J. Fijneman bood tijdens het hele afstudeerproject hulp bij het maken van het verslag, de opzet van bepaalde programmadelen, de aanpak van het onderzoek en een beter begrip van het commentaar.

Verder wil ik de volgende personen bedanken die van belang zijn geweest voor de voortgang of correcte opzet van dit project. Veelal mocht ik gebruikte programmatuur bekijken, boeken lenen, afbeeldingen gebruiken en informatie inwinnen.

Ir. Freide,	Corsmit Den Haag,	konstruktiebureau
Ir. Van Gorp,	Tielemans Eindhoven,	konstruktiebureau
Ir. Beekx,	Van de Laar Eindhoven,	konstruktiebureau
Dr. Dignum,	TU Eindhoven,	faculteit Wiskunde & Informatica
Ir. Stuurstraat,	TU Delft,	faculteit Civiele Techniek
Ing. Renders,	TU Eindhoven,	faculteit Werktuigbouwkunde
Dr. Nederpelt,	TU Eindhoven,	faculteit Wiskunde & Informatica
Ir. Ahn,	KU Brabant,	faculteit Informatica

Mijn ouders en Bart, Edith, vrienden, en leden van KOers toonden belangstelling en leefden mee. Hen ben ik het meest dankbaar.

Herm Hofmeyer

Inhoudsopgave

In separate bundels:

Bijlage Gebruiksaanwijzing, code en toelichting bij programma
Bijlage T8-T9-verslagen

1 Inleiding en doelstellingen

1.1 Inleiding

Bij de plannen voor een gebouw of de aanpassing van een gebouw horen werkzaamheden om de plannen realiteit te laten worden. Uit een programma van eisen zal een produkt moeten ontstaan dat aan de gestelde eisen voldoet. Een deel van de werkzaamheden bestaat uit het vertalen van het programma van eisen naar een ruimtelijke vorm, het ruimtelijk ontwerpen.

Het ruimtelijk ontwerp transformeert een bouwkundig programma van eisen in een ruimtelijke vorm en is lang niet altijd een systematisch proces: ruimtelijk ontwerpers, architecten, zullen meer of minder gebruik maken van creatieve oplossingen voor het programma van eisen. Het programma van eisen kan uiteenlopend van vorm en inhoud zijn en geeft als preconditie al aanleiding tot de meest uiteenlopende processen voor het ruimtelijk ontwerp. Elke ruimtelijk ontwerper heeft zijn eigen ontwerpstrategieen waardoor diverse ruimtelijke ontwerpen aan een programma van eisen kunnen voldoen.

Een andere werkzaamheid is het zodanig ordenen van materialen dat de ruimtelijke vorm in de fysieke wereld kan bestaan, het konstruktief en bouwtechnisch ontwerpen.

Verder zijn er nog vele werkzaamheden zoals het bouwfysische ontwerp en de voorbereiding van de uitvoering en de uitvoering van een bouwproject zelf. De huidige opvatting is dat het aanbevelingswaardig is het ruimtelijk ontwerp en het konstruktief ontwerp met elkaar te laten samenvallen, zodat het ruimtelijk ontwerp wordt afgestemd op het konstruktief ontwerp en evenzo het konstruktief ontwerp wordt afgestemd op het ruimtelijke ontwerp. Een huidige opvatting is tijdelijk en hoeft zeker niet de beste opvatting te zijn. Voordelen van de huidige opvatting zijn de genoemde afstemming, vaak een kostenbesparing, zowel door besparing op materiaal als arbeid, en een esthetisch beter ontwerp. Een nadeel van de huidige opvatting is het feit, dat door het nemen van compromissen, per ontwerponderdeel niet altijd de beste keuze voor dat onderdeel afzonderlijk wordt gemaakt.

1.2 Hoofddoelstelling

Met dit afstudeerproject wordt onderzoek gedaan naar de mogelijkheden om de computer als hulpmiddel te gebruiken bij het konstruktief ontwerp. Hiermee wordt niet de bestaande rol die de computer vervult in het konstruktief ontwerp-proces bedoeld: de rekenaar. De konstrukteur verricht naast ontwerptaken ook diverse rekentaken, zoals het bepalen van het krachten- en/of spanningsverloop in een konstruktie belast door uitwendige en inwendige belastingen, het bepalen van de afmetingen van konstruktie-elementen nadat de belastingen op deze elementen bekend zijn en bijkomende berekeningen. De computer wordt met succes ingezet om diverse genoemde rekentaken van de konstrukteur over te nemen.

Vooral de rekentaken die bestaan uit een telkens terugkerend patroon van bewerkingen zijn hiervoor geschikt. Door het overnemen van deze rekentaken is de computer in de wereld van het konstruktief ontwerp bekend geworden. Nu, tijdens dit afstudeerproject, wordt onderzoek gedaan naar de mogelijkheden om de computer in te zetten bij het werkelijke konstruktief-ontwerpproces. Later in dit onderzoek wordt bekeken wat met konstruktief ontwerpen precies bedoeld wordt. Overigens is niet de computer het werkelijke hulpmiddel dat bedoeld wordt. Een algoritme, een soort recept hoe een probleem kan worden opgelost, wordt door een computer uitgevoerd. Het is een algoritme dat gezocht moet worden om het konstruktief ontwerpen te kunnen ondersteunen en vervolgens wordt dit algoritme uitgevoerd door de computer.

1.3 Nevendoelstelling

De gevonden mogelijkheden (in het T8 en T9-projectwerk, zie de bijlage "T8-T9-verslagen") om het konstruktief ontwerpen te ondersteunen met behulp van een computer staan niet los van andere ontwerpactiviteiten, ook omdat er een samenhang is tussen het ruimtelijk ontwerp en het konstruktief ontwerp. Zo is voor het proces van ruimtelijk ontwerpen een heel scala van programmatuur ontwikkeld, vaak op basis van space-allocation. Space-allocation is het ordenen van ruimten ten opzichte van elkaar op basis van ingevoerde eisen omtrent de ordening van deze ruimten:

Stel dat een persoon drie blokken uit een blokkendoos voorgelegd worden en dat hem gevraagd wordt deze blokken zo te rangschikken dat een esthetisch geheel onstaat. De kans is groot dat bij iedere persoon die deze vraag gesteld wordt een andere rangschikking van blokken ontstaat. Genereert een computerprogramma alle mogelijke varianten van blokkenstapelingen, dan is de kans niet gering dat een persoon bij het aanschouwen van een der varianten tot de conclusie zal komen dat deze variant (door de computer gegenereerd) esthetisch beter is. De persoon had aan deze variant niet gedacht. Door een apparaat allerlei varianten te laten ontwikkelen (door rekenkracht) is de kans aanwezig dat er een variant ontstaat waar de ontwerper niet aan heeft gedacht. Het idee achter deze gegenereerde variant kan door de ontwerper gebruikt en verder uitgewerkt worden.

In dit afstudeerproject wordt onderzocht of het mogelijk is om de techniek space-allocation -een methode om ruimtelijk te ontwerpen- te combineren met de gevonden mogelijkheden om het konstruktief ontwerpen te ondersteunen. In een gunstig geval is het zelfs mogelijk dat deze twee technieken elkaar kunnen versterken en elkaars teveel aan ontwerpmogelijkheden effectief kunnen beperken.

1.4 Doelstellingen als gevolg van de hoofd- en nevendoelstelling

Door het realiseren van de hoofd- en nevendoelstelling zullen ook diverse andere zaken tot doel gesteld worden. Tijdens dit afstudeerproject zal kennis opgedaan worden over hoe konstruktief ontwerpprocessen functioneren, hoe konstruktief en ruimtelijk ontwerpen te

analyseren zijn, de achterliggende structuur van computergebruik, logica, kennissystemen, kennisrepresentatie, de interne opbouw van een computer en diverse andere zaken.

2 Opzet onderzoek

2.1 Inleiding

Om de doelstellingen van dit afstudeerproject te realiseren worden een aantal werkzaamheden uitgevoerd. Globaal zijn deze werkzaamheden in overeenstemming met de richtlijnen die in het vakgebied informatica gelden voor de bouw van een kennissysteem. Deze richtlijnen zijn kort beschreven:

- Kennis opdoen van vakgebied, door middel van interviews, literatuurstudie
- Kiezen voor procedurele of declaratieve aanpak, afhankelijk van probleem
- Bij procedurele aanpak: maak algoritmen en zet deze om in een procedurele taal
- Bij declaratieve aanpak: maak regels die kennis vakgebied beschrijven en voer deze in declaratieve taal in
- Test programma op correctheid (alle mogelijke invoer wordt goed verwerkt), volledigheid (alles wat als uitvoer mogelijk moet zijn is ook mogelijk), binnen vakgebied
- Pas procedures of regels indien nodig aan

Op deze richtlijnen wordt hier niet verder ingegaan. Ze zijn te vinden in diverse bronnen.

2.2 Kennisvergaring

Ten eerste wordt onderzoek verricht naar het proces konstruktief ontwerpen zelf door:

- Interviews met konstrukteurs en ontwerpers te houden
- Literatuur te bestuderen
- Gerealiseerde konstruktief ontwerpen bekijken en analyseren

Ten tweede wordt onderzoek verricht naar de achterliggende structuur van computergebruik door:

- Het bestuderen van de logica
- Het bestuderen van de opbouw en fabricage van kennissystemen
- Het bestuderen van de leer van kennisrepresenatie
- Het bestuderen van de interne opbouw van een computer

Merk op dat het tweede punt (het bestuderen van de opbouw en fabricage van kennissystemen) zelf weer ten dele nodig is om de opzet van het onderzoek van dit afstudeerproject op te zetten.

2.3 Kennis omzetten in processen en data

De verkregen kennis door dit onderzoek wordt geabstraheerd en geformaliseerd in proces- en datamodellen. Procesmodellen beschrijven op een vooraf vastgelegde manier processen. Datamodellen beschrijven de gegevensstromen tussen bepaalde processen op een vooraf vastgelegde manier. In dit afstudeerproject is ervoor gekozen de methode IDEF0 te gebruiken om procesmodellen (te vergelijken met algoritmen, een procesmodel beschrijft een deelproces of proces) te beschrijven. Datamodellen beschrijven de gegevens die in de gegevensstromen en processen gebruikt worden. Om datamodellen te beschrijven wordt de methode IDEF1x gebruikt. Beide methoden, IDEF0 en IDEF1x, worden ook gebruikt in het Bouw-Informatie-Model (BIM) [3]. Zowel het BIM alsmede IDEF0 en IDEF1x worden nog beschreven.

2.4 Processen en data omzetten in algoritmen

Na het verkrijgen van de benodigde kennis door onderzoek, en het omzetten van deze kennis in proces- en datamodellen, wordt deze kennis vertaald in een enkele of meerdere algoritmen. Deze algoritmen moeten, volgens de hoofd- en nevendoelstelling van dit afstudeerproject, het konstruktief ontwerpproces ondersteunen en samenwerken met al bestaande algoritmen voor een methode van ruimtelijk ontwerpen: space-allocation. Hoewel de verkregen kennis door onderzoek gekleurd wordt door de zienswijze van de onderzoeker, is in veel sterkere mate het probleem aanwezig dat de vertaling van kennis naar een enkele of meerdere algoritmen subjectief is. Soms is het al prettig dat er een algoritme gevonden wordt, ook al voldoet deze maar ten dele, in een ander kennisgebied houdt men zich alleen bezig met het perfectioneren en kiezen van algoritmen.

2.5 Een werkend programma

In laatste instantie wordt deze gevonden algoritme(n) verwerkt tot een werkend programma. Hoewel het feit dat een programma maken veel arbeid kost en er vaak nog weinig functioneel belang voor het programma bestaat, is een werkend programma maken belangrijker dan in eerste instantie gedacht wordt. Er ontstaan bij de fabricage van het programma concrete problemen die bij het opzetten van een algoritme over het hoofd gezien worden. Hierdoor is het mogelijk het algoritme bij te schaven en de zekerheid te hebben dat het algoritme werkelijk functioneert. Het realiseren van een programma maakt het niet alleen mogelijk dat werkende algoritmen gevonden worden, maar ook dat getoetst kan worden of de algoritmen de hoofd- en nevendoelstellingen van dit afstudeerproject mogelijk maken: een programma toetst het idee.

Om bij dit afstudeerproject de algoritme(n) uit te voeren is gekozen voor de declaratieve programmeertaal "Prolog-2 voor Windows 3.1". Een declaratieve programmeertaal werkt met de invoer die het probleem beschrijft en vindt hierna zelf de oplossing, dit in tegenstelling tot een procedurele taal die met de invoer werkt hoe een probleem opgelost dient te worden.

2.6 Evaluatie, realiseren van de doelstellingen

Op het einde van het afstudeerproject wordt bekeken in hoeverre de doelstellingen gerealiseerd zijn. Een belangrijke conclusie is dat het echte creatieve konstruktief ontwerp niet vervangen kan worden door een algoritme, iets dat ook gevoelsmatig onderstreept kan worden. Het is echter wel mogelijk sommige taken die nu tot het domein van de konstrukteur behoren over te nemen door een algoritme uitgevoerd door een computer.

3 Kennisvergaring

3.1 Literatuur

Er wordt onderzoek gedaan naar het proces konstruktief ontwerpen zelf door interviews te houden met konstrukteurs en ontwerpers, door gerealiseerde konstruktieve ontwerpen te bekijken en te analyseren en door literatuur te bestuderen. Als literatuur zijn de volgende bronnen bestudeerd.

3.2 Building Structures

In het boek van Ambrose, Building Structures, wordt een indeling van konstrukties gemaakt evenals enige eigenschappen van deze klassen van konstrukties belicht [1]. Dit dient als voorbeeld voor een indeling in klassen. In andere literatuur worden weer andere indelingen gevonden. Konstrukties kunnen in drie hoofdklassen worden onderscheiden:

Solid structures
- Sterkte en stijfheid door massa
- Erg goed bestand tegen dynamische belastingen
- Rekenen is erg moeilijk

Surface structures
- Wand en drager
- Stabiel en sterk
- Geconcentreerde lasten moeilijk opneembaar

Framed structures
- Vormflexibel
- Dragen speciale belastingen goed mogelijk

Constructies kunnen ook verdeeld worden in meerdere, wat kleinere, klassen:

Post-Beam
- Kolommen en balken systeem
- Invulling vereist een aanvullende constructie

Rigid frames
- Een Post-Beam constructie met stijve verbindingen.
- Verbindingen moeilijker te maken dan bij Post-Beam constructies.

Plaatsystemen

- Handig bij overspanningen in meer richtingen. Wanneer de lengte kleiner wordt dan twee maal de breedte is er nauwelijks nog voordeel te bespeuren bij een plaatsysteem.

Vakwerken
- Bewerkte balken
- Verhoogde efficiëntie
- Sluit aan bij Post-Beam systemen

Boog, cilinder en dome structuren
- Veel in beton
- Berust op drukbogen, dus geen buiging

Surface structures
- Vaak niet-repeteerbare constructies
- Krachten werken in het vlak van het materiaal

Special systems
- Blaashal
- Geodesic domes

3.3 Bauwerk Tragwerk Tragstruktur

Bauwerk Tragwerk Tragstruktur is een verzameling van 2 banden die tezamen het konstruktief ontwerpen proberen te systematiseren [2]. De eerste band analyseert de natuurlijke en de gebouwde omgeving. De tweede band klassificeert de verschillende draagstrukturen en behandelt de draagkwaliteit. Verder staan in de tweede band ook veel voorbeelden van konstruktieve ontwerpen. In de eerste band is kort beschreven hoe het ontwerpen van draagstrukturen te systematiseren is volgens de auteurs. Het konstruktief ontwerpen is deel van het totale ontwikkelingsproces van een gebouw maar ook een tijdsgebonden deel van het ontwerpproces. Dat wil zeggen dat een groot gedeelte van het konstruktief ontwerpen pas na een aantal andere ontwerpprocessen kan beginnen en dat wederom andere ontwerpprocessen pas kunnen starten op het moment dat het konstruktief ontwerp gereed is. In de afgelopen jaren is over de hele wereld en door diverse mensen geprobeerd het ontwerpproces te systematiseren. Een voorbeeld hiervan is te zien in de tweede afbeelding in figuur 1. Deze illustratieve afbeeldingen zijn op het einde van dit verslag groter afgedrukt in een bijlage. Bij sommige deelprocessen van het ontwerpproces ging dat prima terwijl bij andere deelprocessen het heel wat moeilijker ging. De problemen komen vaak omdat bij deze deelprocessen:

- de scheppende rol van de mens in het ontwerpproces niet gezien of genegeerd wordt,

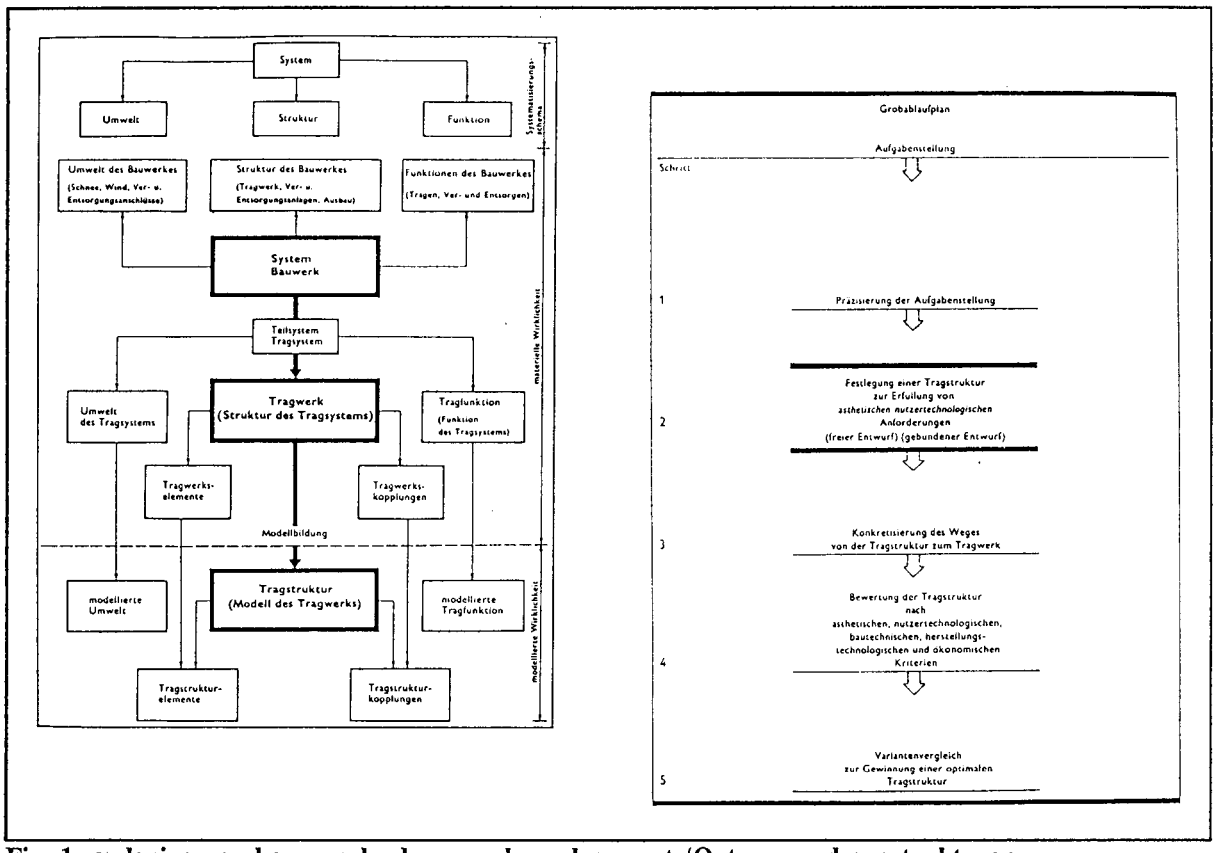


Fig. 1, ordening van bouwwerk, draagwerk en draagsyst./Ontw. van draagstrukturen

- de mogelijkheid van de mens om combinaties van kwalitatieve eisen te kunnen verwerken niet gezien wordt,

- hulpmiddelen ter systematisatie gezocht worden die reeds succesvol zijn bij de analyse van ontwerpen (bijvoorbeeld rekenapparatuur).

"Es ist abzusehen, daß in allernächster Zukunft die Bestimmung des Schöpferischen im Entwurfsprozeß neuen Inhalt erfahren kann und mancher Vorgang, der heute noch als schöpferische Äußerung menschlicher Intuition empfunden wird, als Lösung einer formalisierbaren Entwicklungssituation der Maschine übertragen wird. Immer aber wird der Mensch unentbehrlichter Gestalter des Entwurfsprozesses bleiben."

"Het is te verwachten, dat in de nabije toekomst het gebruik en de beheersing van het beeldend vermogen in het ontwerpproces een nieuwe dimensie krijgt en veel vooruitgang, die nu nog als scheppende uiting van de menselijke intuitie gezien wordt, door middel van een algoritme aan de machine wordt overgelaten. De mens zal echter altijd de bron achter het ontwerpproces blijven."

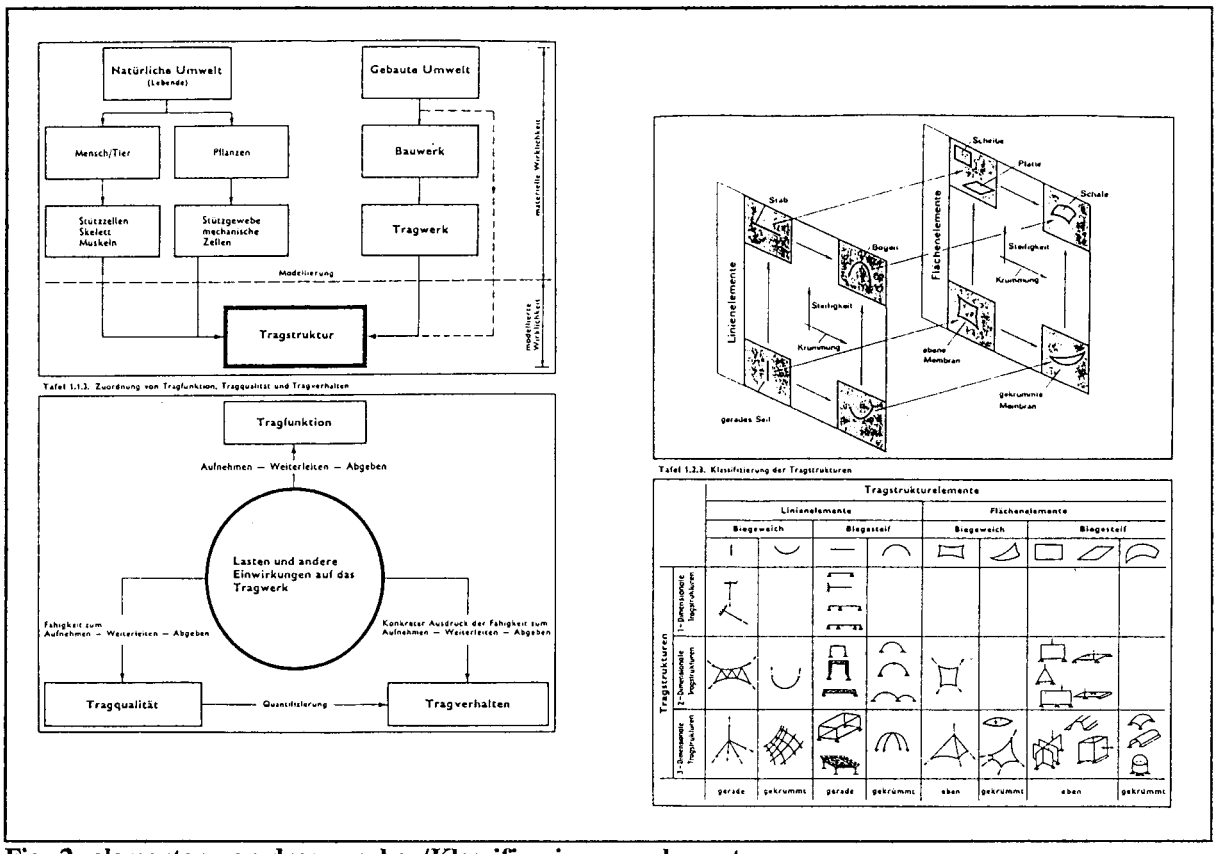


Fig. 2, elementen van draagwerken/Klassificering van elementen

Het konstruktief ontwerpen is reeds van belang aan het begin van het ontwerpproces maar de invloed van een eenmaal gekozen hoofddraagstruktuur is tot het einde toe belangrijk. Verder heeft een gekozen hoofddraagstruktuur veel invloed op de funktionele en esthetische kwaliteiten van een gebouw. Het konstruktief ontwerpen kan dankbaar gebruik maken van een aantal rekentechnieken hoewel dit altijd slechts hulpmiddelen blijven ter ondersteuning van de menselijke creativiteit. Het konstruktief ontwerp-proces kan opgedeeld worden in 5 delen. In de eerste plaats moet het programma van eisen en het ruimtelijk ontwerp zo vertaald worden, dat randvoorwaarden en uitgangspunten voor een konstruktiekeuze voorhanden zijn. Deze randvoorwaarden en uitgangspunten kunnen dienen, in de tweede plaats, om een aantal principieel verschillende konstruktieve oplossingen te genereren. Deze konstruktieve varianten worden dan in zoverre uitgewerkt (stap 3) dat een beoordeling van de technische-, omgevings- en esthetische kwaliteiten van de varianten kan worden gegeven. Op het einde (stap 5) vindt dan een uiteindelijke keuze plaats voor een konstruktieve ontwerpvariant. In figuur 1, tweede afbeelding, is deze voorgenoemde werkwijze te zien en op de eerste afbeelding van figuur 4 is een nauwkeuriger stappenplan om het ontwerp te systematiseren gegeven. In een bijlage van dit verslag zijn deze afbeeldingen vergroot weergegeven. De auteurs wijzen erop dat de voorgestelde systematiek slechts een veelvoud van ontwerpprocessen beschrijft, maar niet voorschrijft dat een

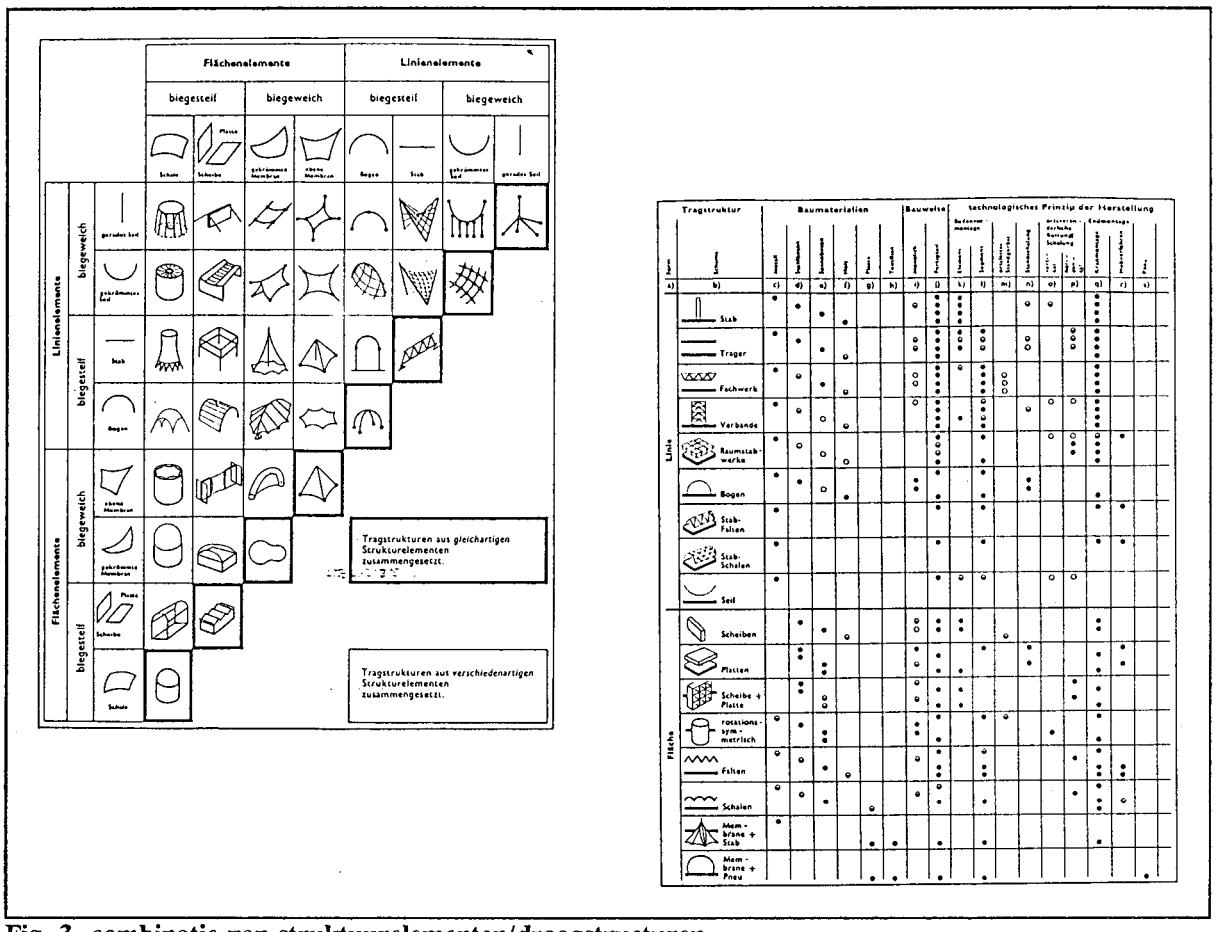


Fig. 3, combinatie van struktuurelementen/draagstructuren

willekeurig ontwerpproces zo moet geschieden. Om het proces zoals het nu beschreven is werkelijk te kunnen toepassen is kennis nodig. Kennis van de verschillende bekende draagstrukturen, die een schematisatie vormen van de fysieke konstruktie. Kennis van de draagkwaliteit van draagstrukturen zodat men weet hoe globaal draagstrukturen werken (het afgeven van krachten, momenten, druk, knik en trek, enz.). Er zijn overzichten nodig van draagstrukturen, geordend naar functie en geometrie. Verder is het nodig te weten hoe de draagkwaliteit gekwantificeerd kan worden tot het zogenaamde "tragverhalten", vrij vertaald met "draagkracht".

In deel 2 van Bauwerk Tragwerk Tragstruktur worden de verschillende begrippen nog eens goed omschreven. In de eerste afbeeldingen van figuur 1 en figuur 2 zijn deze begrippen in hun verband te zien. De fysieke konstruktie, in het boek aangegeven met Tragwerk, kan worden verdeeld in elementen van een bepaald materiaal en een bepaalde geometrie. De elementen van de draagstruktuur, de geschematiseerde fysieke draagkonstruktie, kunnen ook naar geometrie en materiaal worden onderverdeeld, zij het met behulp van andere attributen per entiteit materiaal en geometrie. De elementen van de

draagstruktuur kunnen dan worden geklassificeerd door onderscheid te maken tussen lijnvormige elementen en plaatvormige elementen waarbij de kromming en stijfheid kan varieren. Als de elementen van de draagstruktuur geklassificeerd zijn, zijn ook de draagstrukturen zelf te klassificeren. Draagstrukturen kunnen opgebouwd worden uit 1 elementtype, maar bijna vanzelfsprekend ook uit meerdere elementtypen zoals op de tweede

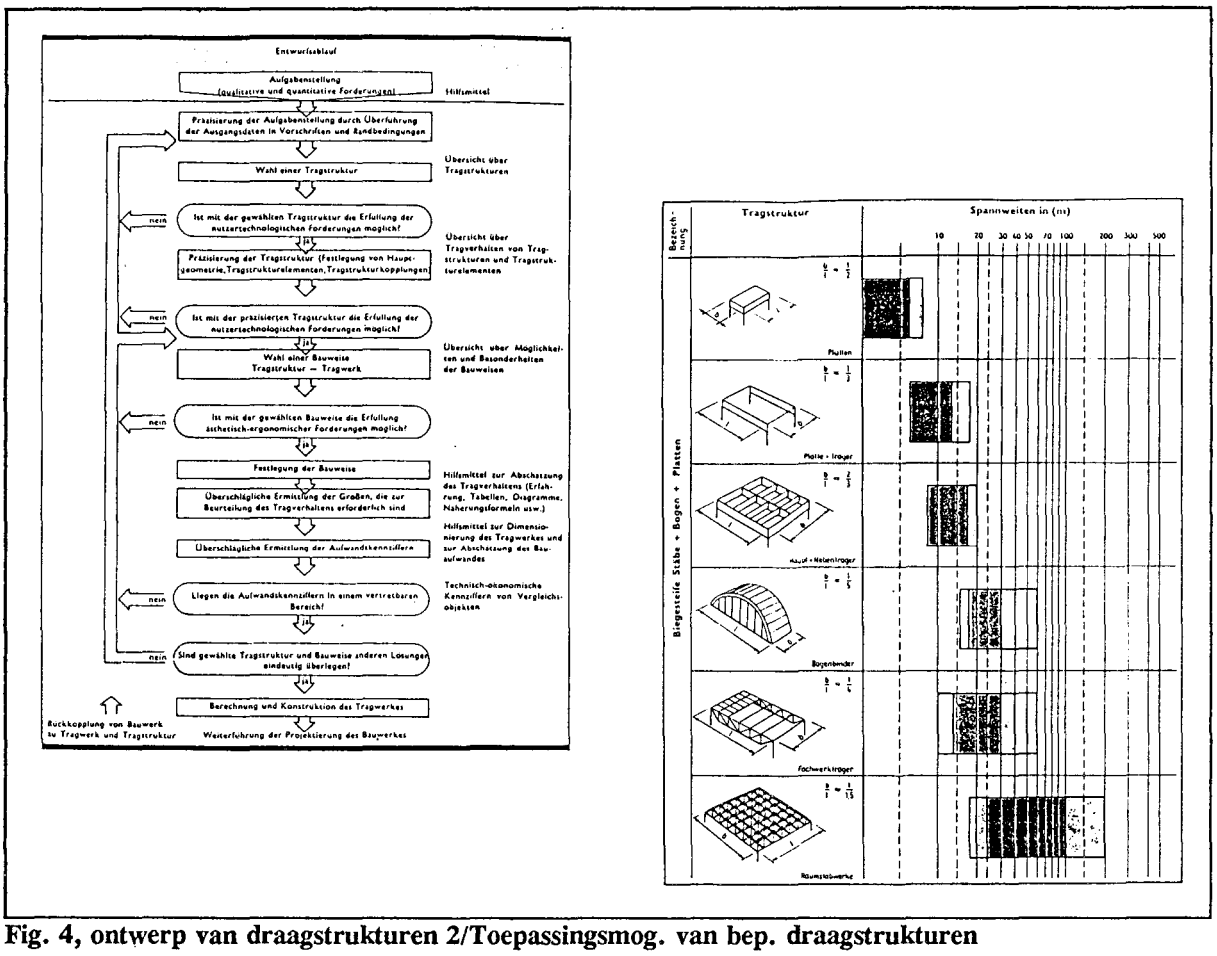


Fig. 4, ontwerp van draagstrukturen 2/Toepassingsmog. van bep. draagstrukturen

afbeelding van figuur 2 en de afbeeldingen van figuur 3 te zien is. Wat betreft de wisselwerking tussen bouwwerk en de fysieke draagkonstruktie, worden gebouwen in deel 2 ingedeeld in 3 verschillende klassen. Klasse 1 behelst de gebouwen die een fysieke draagkonstruktie kennen die alleen de dragende functie vervult en verder niets doet. De konstruktie bepaalt in principe niet de vorm van het gebouw en zorgt vaak ook niet voor de begrenzing van ruimten. De tweede klasse van gebouwen bestaat uit gebouwen waarbij de konstruktie duidelijk de vorm van het gebouw bepaald, zoals bij daken bijvoorbeeld bestaande uit cylinderschalen, koepels, enz. Deze klasse van gebouwen komt vaak voor als grote ruimten kolomvrij horen te zijn. De gebouwen zijn vaak flexibel in te delen wat betreft de binnenindeling. Ook gebouwen met een stabiliteitskern behoren volgens de

auteurs tot deze klasse. De derde klasse van gebouwen heeft als kenmerk dat de konstruktie bijna alle functies van het gebouw verwerkt. Gedacht kan worden aan bruggen, sportcomplexen in de vorm van ballonhallen, schoorstenen, silo's en koeltorens. In de tweede afbeelding van figuur 4 staat een gedeelte van een tabel (als illustratie) waarbij bij een gegeven overspanningsmaat mogelijke draagstrukturen te vinden zijn. In de tweede afbeelding van figuur 3 zijn vervolgens bouwmateriaal en fabricagegegevens beschikbaar.

3.4 De analyse van 4 konstruktieve ontwerpen/interviews konstrukteurs

Bij de analyse van konstruktieve ontwerpen is het verstandig van te voren te weten waar bij de analyse de nadruk op gelegd wordt. Hoe wordt de analyse uitgevoerd en hoe is het mogelijk om de verschillende ontwerpen met elkaar te kunnen vergelijken of tegen elkaar te kunnen afzetten is een belangrijke vraag.

Er moet begonnen worden met een globale planverkenning en een globale konstruktieverkenning van het te analyseren plan om het plan te kunnen ordenen en te kunnen structureren. Vragen zijn of het plan gestructureerd opgebouwd is of juist niet. Een gestructureerd plan (een plan met een pragmatisch ontwerpproces tot stand gekomen) leent zich sterk voor het onderzoek naar hoe van een ruimtelijk plan naar een konstruktief plan kan worden gegaan. Een minder gestructureerd plan helpt inzichten te verkrijgen naar de omzetting van een programma van eisen naar een ruimtelijk plan.

Vervolgens is er een onderscheid te maken naar plannen die vanuit het programma van eisen ineens omgezet zijn in een vorm en een konstruktie en naar plannen die een duidelijke scheiding te zien geven tussen het ruimtelijk ontwerp en het konstruktief ontwerp. Zonder een oordeel over deze 2 planontwerpbenaderingen te geven is duidelijk dat de eerste planaanpak met de huidige opvattingen beter strookt hetgeen echter niet wil zeggen dat deze benadering ook werkelijk kwalitatief beter is.

In het algemeen is de procesgang van het ontwerpproces belangrijk. Hoe is het proces onstaan, hoe verliep het proces en wat is het resultaat van de procesgang voor het ontwerp, zouden vragen zijn die gesteld moeten worden.

De materiaalkeuze en konstruktiekeuze zijn te specificeren bij het oplossen van de bovenstaande vragen.

In het boek Bauwerk, Tragwerk, Tragstruktur vinden we de vragen, volgens de auteurs, die we moeten stellen bij de analyse van de gebouwde omgeving en dan met name bij de analyse van konstruktieve ontwerpen. Deze vragen kunnen helpen een voorzichtige conclusie te trekken over het ontwerpproces:

Is het, bij het zien van het bouwwerk, direct mogelijk de fysieke konstruktie te onderscheiden?

Is het, bij het zien van de fysieke konstruktie, direct mogelijk een schematisatie van de konstruktie te maken?

Is de vertaling van de fysieke draagkonstruktie naar een schematisatie slechts door grote idealiseringen mogelijk?

Is de fysieke draagkonstruktie niet op meerdere manieren te schematiseren?

Zorgt de gekozen draagstructuur voor de gunstigste draagsituatie in het bouwwerk?

Is de fysieke draagkonstruktie, met de bouwmethode in beschouwing genomen, het meest doelmatig ontworpen met de draagstructuur als achterliggend schema?

Sluit de gekozen konstruktie aan bij de stijl van het gebouw?

3.4.1 De Centrale in Den Haag van konstruktiebureau Corsmit

Bij konstruktiebureau Corsmit in Den Haag is gesproken met Jan Font Freide over het ontwerp De Centrale, eveneens in Den Haag:

Gebouw: De Centrale in Den Haag
Architect: Kohn Pendersen Fox, Londen
Konstrukteur: Corsmit, Den Haag

Globale planverkenning en globale konstruktieverkenning

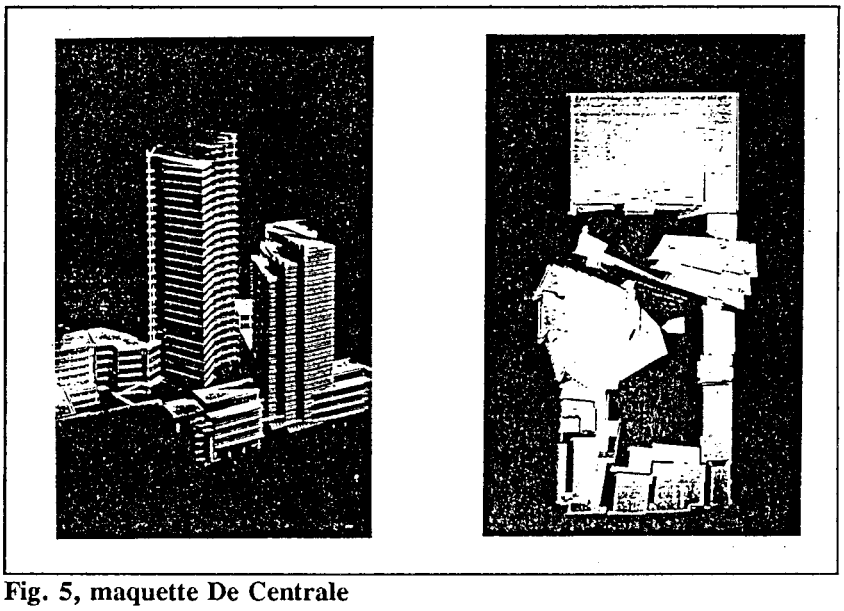


Fig. 5, maquette De Centrale

De Centrale bestaat op dit moment als een plan in bestekfase, uitgewerkt in een bestek en bestektekeningen. Een maquette is te zien in figuur 5. Het plan bestaat in hoofdzaak uit 2 grote kantoortorens, enige laagbouw en een serre, ingebed tussen bestaande bebouwing. De genoemde laagbouw bestaat uit circa 4 tot 5 verdiepingen. De serre verbindt de twee kantoortorens en een deel van de laagbouw. Alleen de grote kantoortoren wordt besproken. De grote kantoortoren bestaat uit ongeveer 25 lagen en is vormgegeven als 4 in dikte verlopende schijven die tegen elkaar aanliggen. De konstruktie werkt niet mee aan de expressie van het gebouw omdat de konstruktie bestaat uit een betonnen kern met plaat-

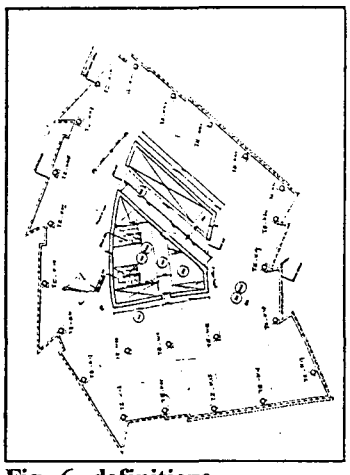


Fig. 6, definitieve plaatvloer De Centrale

vloeren waarbij kolommen in de gevel helpen om de vloer te dragen. Van buiten gezien suggereert de vorm van de grote kantoortoren echter dat de konstruktie ook uit 4 schijven (in dikte verlopend) bestaat. De plaatvloer, te zien in figuur 6, is een getand exemplaar: langs de randen zijn tanden gemaakt om de gevelvorm te verkrijgen zoals die door de architect gewenst is. Het ontwerp is ruimtelijk pragmatisch (een toren met x verdiepingen) en is als zodanig goed geschikt (zie punt 1.2.2) om de translatie van ruimtelijk ontwerp naar konstruktief ontwerp te bekijken. Dit ontwerp is eerst als ruimtelijk ontwerp gemaakt waarna een konstruktie werd ontworpen. De konstrukteur is wel vroegtijdig bij het ontwerp betrokken geraakt, maar de konstruktie doet niet mee aan de expressie van het gebouw. Volgens de konstrukteur is deze werkwijze de meest voorkomende. De konstruktie bij dit gebouw hoort wel bij het gebouw, maar versterkt de expressie van het gebouw niet. Het zou in de ogen van velen mooier zijn een konstruktie te maken bestaande uit schijven die er net zo uit zien als het gebouw. Volgens de konstrukteur is er dan geen stabiliteit te verkrijgen en is een konstruktie met een kern gemeengoed in Nederland.

Procesgang

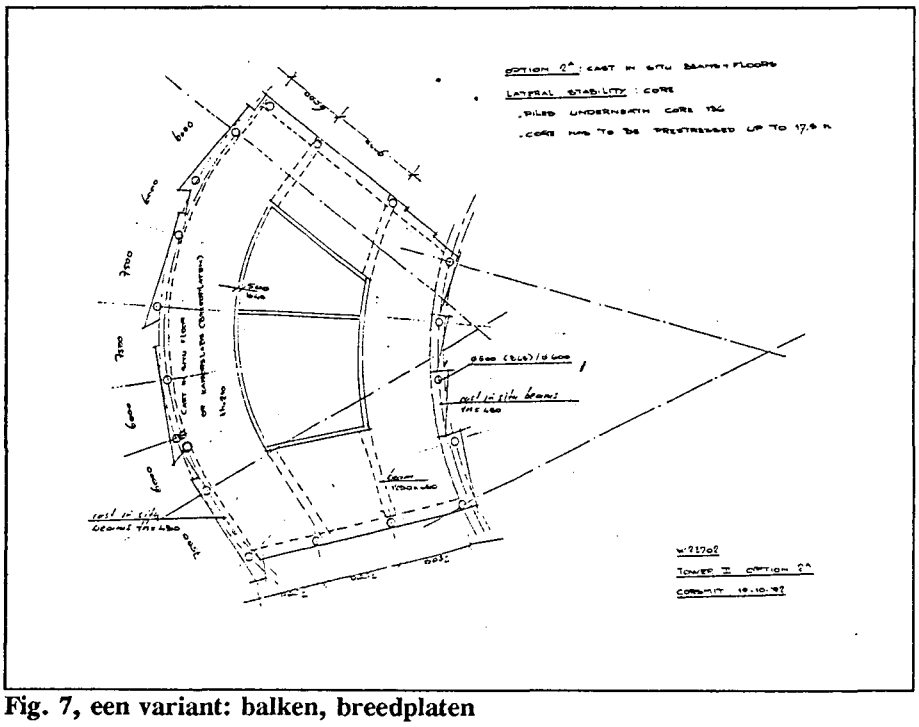


Fig. 7, een variant: balken, breedplaten

De opdrachtgever van De Centrale wilde 55000 vierkante meter bedrijfsoppervlak, 450 parkeerplaatsen en 2 onderliggende lagen ten dienste van deze parkeerplaatsen. Na een bezoek aan de gemeente Den Haag bleek een locatie beschikbaar te zijn. Met een schets werd de opdrachtgever duidelijk gemaakt dat er een plek was waar een hoog gebouw mocht staan en er

een plek was waar een lager gebouw mocht staan. Verder was op de locatie bestaande bebouwing aanwezig en nog wat ruimte vrij om laagbouw op te situeren. Met deze schets ging de opdrachtgever naar de architect: een "vorm"-architect. Deze architect maakte een plan in Londen dat wordt afgekeurd door de opdrachtgever. In kort bestek werd een nieuw kleimodel gemaakt voor De Centrale en dit model werd goedgekeurd door de gemeente en opdrachtgever. Hierna maakte de architect goede tekeningen en vroeg de konstrukteur om advies. De konstrukteur maakte een aantal globaal konstruktieve varianten zoals deze te zien zijn op de afbeeldingen. De konstrukteur koos omdat het om een grote toren handelde en in Nederland gebruikelijke bouwmethoden gezocht werden voor de volgende konstruktieve hoofddraagprincipes: een kern met een vloer op balken, te zien in figuur 7, een kern met plaatvloer, een kern met een vloer op dwarsbalken en een kern met kanaalplaten. Konstruktieve hoofddraagprincipes zonder kern werden niet in beschouwing genomen: de aanwezigheid van een kern werd door de konstrukteur als vanzelfsprekend ervaren. Een paar konstruktieve varianten vielen af omdat de architect zich niet kon verenigen met de consequenties van zo'n konstruktie. Zo zou een dragende gevel kleine ramen tot gevolg hebben gehad en dat wilde de architect onder geen beding. Een ander voorbeeld is dat de architect geen zogenaamde "outrigger", te zien in figuur 8, wilde omdat dat esthetisch niet goed bevonden werd. Andere globaal konstruktieve varianten vielen af omdat deze te duur waren in vergelijking met de gekozen variant. Tijdens de keuze voor een konstruktieve variant van de hoofddraagkonstruktie waren een kostendeskundige, de opdrachtgever, de architect, de bouwfysicus, de konstrukteur en een installatiedeskundige aanwezig. Bijna altijd wordt volgens de konstrukteur gekozen voor de goedkoopste variant van een hoofddraagkonstruktie die nog net voldoet aan de eisen gesteld door installatievoorzieningen (en andere bouwkundig technische randvoorwaarden) en die de mogelijkheden biedt die de architect wenst zoals te zien was in het voorbeeld van een dragende gevel die een "open" vormgeving belette. Nadat de

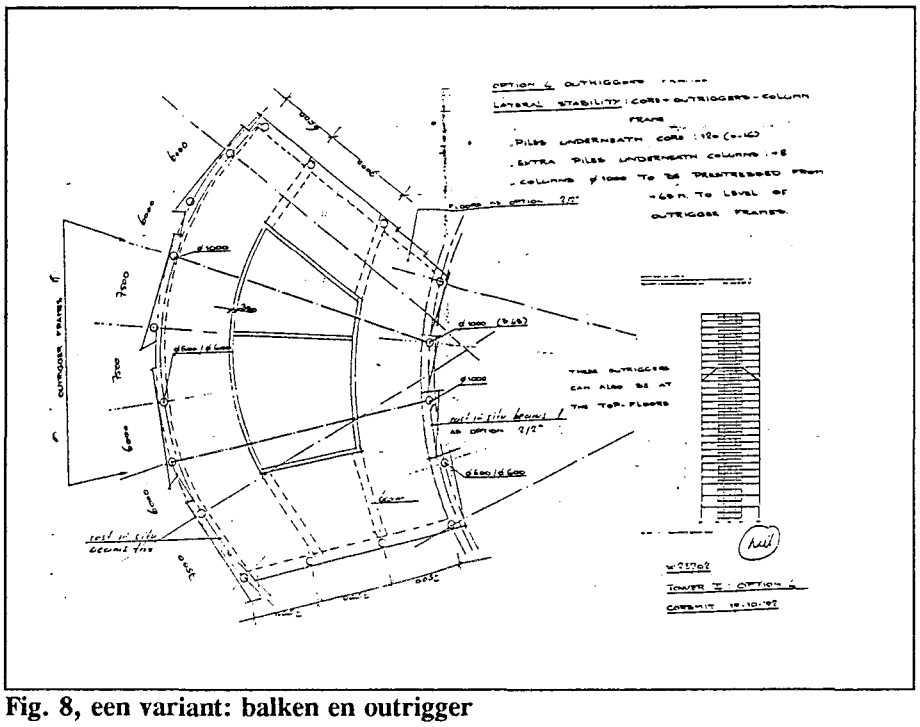


Fig. 8, een variant: balken en outrigger

keuze voor de hoofddraagkonstruktie gemaakt was, kwam de kolomplaatsing aan de orde. Nadat de konstrukteur een kolomplaatsing had voorgesteld kwam de architect met wijzigingen. De uiteindelijk indeling van de systeemplafonds had zelfs invloed op de kolomplaatsing. Ook de "punten" aan de vloer waren te groot om zonder grote vervormingen te kunnen worden gerealiseerd: de konstrukteur bood twee alternatieven om dit probleem aan te pakken: een kleinere punt zodat minder doorbuiging onstaat of een hulpkonstruktie onder de punten waardoor de vervorming voorkomen wordt. De architect kon zich niet verenigen met het tweede alternatief en zodoende werden de punten kleiner gemaakt. Behalve de twee torens werd ook laagbouw ontworpen bij de torens en een serre. Voor de laagbouw werd een hoofddraagkonstruktie gekozen die verwantschap had met de hoofddraagkonstruktie van de kantoortorens om uitvoeringstechnische redenen. Hierna werd een kolommensysteem gekozen met de voorwaarde dat de architect kolommen uit de hoek wilde, waarna de konstrukteur zodanig konstrueerde dat dit mogelijk was. Bij de serre of glaskap ziet men de konstruktie en de architect liet de konstrukteur dan ook de vrije hand om iets te ontwerpen. Er is gekozen voor drie-dimensionale vakwerkliggers. Het blijkt dat het rekenwerk van de konstruktieberekeningen ongeveer een vierde van het papierwerk vraagt ten opzichte van de ontwerpvragen en in dit geval ontwerpfaxen van de architect. De faxen die de architect stuurde met vragen over het ontwerp (in de vorm van: "Kan dit wel of niet?, is deze overstek nog mogelijk?") zijn volgens de konstrukteur nooit door een computer te begrijpen.

Vragen uit Bauwerk, Tragwerk, Tragstruktur

Is het, bij het zien van het bouwwerk, direct mogelijk de fysieke konstruktie te onderscheiden?

Nee, de fysieke konstruktie is ingepakt in afwerkingsmaterialen.

Is het, bij het zien van de fysieke konstruktie, direct mogelijk een schematisatie van de konstruktie te maken?

Als de afwerkmaterialen niet aanwezig zijn, is te zien dat het gebouw bestaat uit een kern met vloeren en kolommen in de gevel. Als men vervolgens naar de onderkant van een vloer kijkt, is te zien dat er sprake is van een plaatvloer. Hoe de vloeren gekoppeld zijn aan de kern is in de ruwbouw te zien. Bij de kolommen wordt ervan uitgegaan dat deze zich als pendelstaven gedragen tussen de vloeren.

Is de vertaling van de fysieke draagkonstruktie naar een schematisatie slechts door grote idealiseringen mogelijk?

Nee, het principe van een kern met een plaatvloer is direct als zodanig te schematiseren.

Is de fysieke draagkonstruktie niet op meerdere manieren te schematiseren?

Ja, eventueel kunnen de kolommen meewerken aan de buigstijfheid van het gebouw. Hier is in de berekening ook rekening mee gehouden.

Zorgt de gekozen draagstructuur voor de gunstigste draagsituatie in het bouwwerk?

Dat is erg moeilijk te beoordelen. Volgens de konstrukteur is aan deze vraag voldaan.

Is de fysieke draagkonstruktie, met de bouwmethode in beschouwing genomen, het meest doelmatig ontworpen met de draagstructuur als achterliggend schema?

Er is veel aandacht en overleg geweest in de bestekfase van het gebouw. Volgens de konstrukteur pleegden vooral bouwtechnische adviseurs en konstrukteurs nauw overleg.

Sluit de gekozen konstruktie aan bij de stijl van het gebouw?

Nee, zoals al eerder toegelicht doet de expressie van het gebouw een geheel andere konstruktie vermoeden dan die daadwerkelijk aanwezig is.

3.4.2 Stadskantoor Leeuwarden

Gebouw: Stadskantoor Leeuwarden
Architect: EGM
Konstrukteur: Corsmit, Den Haag

Globale planverkenning en globale konstruktieverkenning, proces

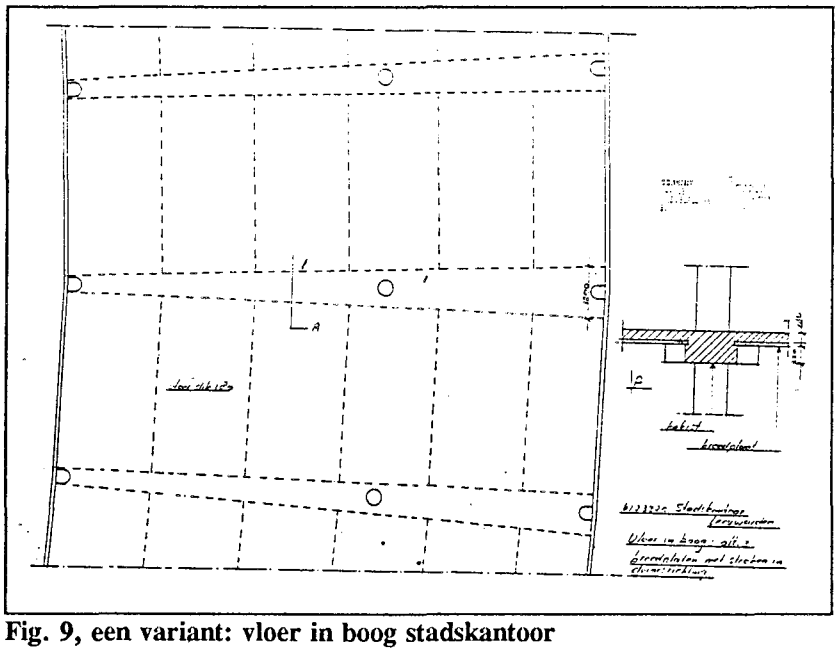


Fig. 9, een variant: vloer in boog stadskantoor

Dit gebouw werd door de konstrukteur aangedragen als gebouw waar de konstruktie wel mee doet aan de expressie van het gebouw. Een indruk van het gebouw kan verkregen worden door afbeelding 10. Het programma van eisen werd bij de voorbereiding nauwkeurig vastgelegd maar het eindresultaat van het ontwerpproces beantwoordt niet precies aan dit programma van eisen: het programma werd bijgesteld. Het stadskantoor moest op een terrein komen waar bestaande bebouwing aanwezig is. Bepaalde

gebouwen van deze bestaande bebouwing werden gesloopt, andere gebouwen bleven staan. Het ontwerp van het stadskantoor werd dus "ingepast" in bestaande bebouwing. Het kantoor moest ongeveer 650 werkplekken bevatten en de mogelijkheid bieden tot het parkeren van 120 auto's. Een aantrekkelijke fietsenstalling voor 225 fietsen en een kantine met 150 zitplaatsen zijn andere gestelde eisen. Een interessant aspect was dat specifieke eisen gesteld werden aan de herkenbaarheid en hoofdvorm van het ontwerp. Het stadskantoor moest als zodanig herkenbaar zijn en niet verward kunnen worden met een willekeurig bedrijfskantoor. De uitwendige verschijning moest extrovert en uitnodigend zijn en de verschillende diensten die in het gebouw gehuisvest worden, mogen als blokken herkenbaar worden gemaakt. Er mocht geen sprake zijn van hoogbouw met een maximale bouwhoogte van 17 meter. Het gebouw werd geacht geschikt te zijn voor een herindeling en baliefuncties hoorden zoveel mogelijk op de begane grond thuis. Het gebouw moest gebouwd worden op de rooilijnen en op specifieke punten zijn ingangen vereist. Er moest sprake zijn van een cellenkantoor met kantoren van minimaal 2,70 meter breed, en 3,50, 5,50, 7,50 en 12,50 meter diep. Het gebouw dat ontworpen is bestaat uit een carré en een schijf die verbonden zijn met een hal, zie hiervoor de afbeeldingen. Zoals ook bij De Centrale het geval was, werden hier eerst konstruktieve hoofddraagvarianten ontworpen waarna hieruit een keuze gemaakt werd. Eén van deze varianten is te zien in figuur 9. In dit geval werd besloten te kiezen voor elementvloeren om esthetische redenen. De elementen gevuld met acoustisch materiaal blijven namelijk na afbouw van het gebouw in het zicht. Volgens

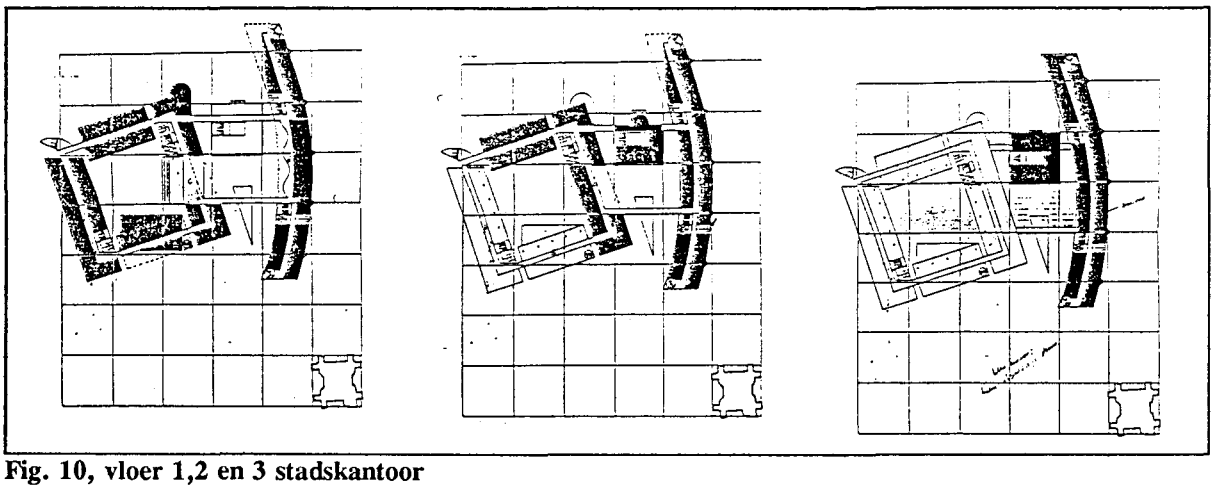


Fig. 10, vloer 1,2 en 3 stadskantoor

de konstrukteur is de rest van het proces vergelijkbaar met de situatie bij De Centrale en behoefde geen nadere uitleg. Dit project is ook niet zo uitvoerig beschreven. De Centrale was het ontwerp van bespreking. Dit ontwerp van het stadskantoor werd aangehaald om te laten zien dat de keuze voor een hoofddraagkonstruktie ook anders gemaakt kan worden en om het feit dat bij dit ontwerp de konstruktie mede bepalend is voor de expressie van het gebouw.

3.4.3 Nieuwbouw Nolte mastenfabriek te Maarheeze

Gebouw: Nolte mastenfabriek te Maarheeze
Architect: H.G.J.A. Manning, architekt hbo-bna, Budel
Konstrukteur: Tielemans, Eindhoven, ir. Van Gorp

Globale planverkenning en globale konstruktieverkenning

Nolte mastenfabriek is een fabriek die onder andere straatlantaarnmasten maakt. Om deze lampenmasten te maken was een grote productiehal nodig van 25 bij 130 meter, met 7 meter vrije hoogte. Verder was er een kantoorgedeelte nodig van circa 43 bij 15 meter en 3 meter hoog. Om de twee gedeelten van productiehal en kantoor met elkaar te verbinden was een tussengedeelte nodig dat diende om ruimte te bieden aan een gang, installaties, toiletten en bergingen. Alle drie de gedeelten moesten gerealiseerd worden als laagbouw. Op de tekening is een indruk te krijgen van de plattegrond van het ontwerp. De productiehal werd uitgevoerd in staal evenals het tussengedeelte. Het kantoor werd uitgevoerd in dragend metselwerk met een betonnen dak.

Procesgang

De architect kwam met een zuiver ruimtelijk ontwerp. Alle wanden waren aangegeven door enkele lijnen en de bedoeling was in samenspraak met de konstrukteur tot een oplossing te komen voor het konstruktief ontwerp, te zien in afbeelding 11. Omdat de architect natuurlijk wel vermoedde dat de grote productiehal van Nolte mastenfabriek in staal gemaakt zou worden, dacht hij het kantoor uit stalen kolommen op te bouwen. Op het kantoor was een betonnen dak onderdeel van het programma van eisen omdat waar mensen werken en toch stil zitten (zoals op de meeste kantoren) een goed klimaat erg belangrijk is. Door een betonnen dak wordt de energie van zoninstraling gebufferd door de grote massa van een betonnen dak. De architect dacht dat stalen kolommen handig waren om dit dak te dragen en gebruikte verder veel metselwerk voor de invulling van de gevels. Toen de konstrukteur dit vele gebruik van decoratief metselwerk zag, onstond het idee om de stalen kolommen te laten vervallen en dragende gevels van metselwerk te ontwerpen die het betonnen dak zouden dragen. Als dak werd gekozen voor holle, voorgespannen kanaalplaatvloeren omdat deze platen snel op het werk te plaatsen

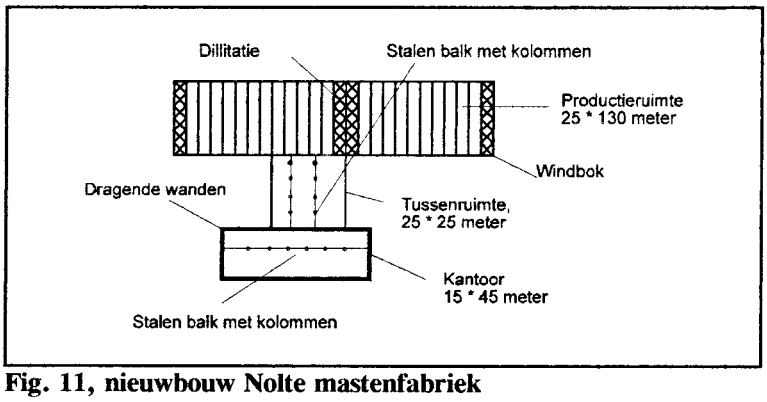


Fig. 11, nieuwbouw Nolte mastenfabriek

zijn. Of deze platen ook als konstruktieve oplossing het beste waren is niet onderzocht: de tijdsbesparing door de toepassing van de gekozen platen was het belangrijkste. Uit ervaring noemde de konstrukteur het gegeven dat de overspanningsmaten bij staal altijd erg economisch zijn in de orde van 5 of 5,5 meter. Bij beton is een goede overspanningsmaat bereikt bij ongeveer 7,2 meter. Omdat het kantoor 15 meter breed is, is een overspanning in 1 keer met een betonnen plaat niet zinvol. Een stalen balk ongeveer in het midden van het kantoor in de langsrichting ondersteunt de betonnen dakplaten. Onder deze balk staan kolommen op ongeveer 5,5 meter omdat dit voor staal zoals genoemd een zinvolle overspanningsmaat is. De productiehal was een hal van 25 bij 130 meter en 7 meter hoog en moest geheel kolomvrij zijn. Deze hal in beton uitvoeren kostte veel meer geld dan een uitvoering is staal. Zou de productiehal moeten bestaan uit 2 lagen dan was een uitvoering in staal met een betonnen vloer interessant, zelfs indien volkomen kolomvrije ruimten vereist werden. Bij 3 lagen en meer zou men aan beton moeten denken als materiaal voor de gehele konstruktie van een ruimte van 25 bij 130 meter. De architect dacht aan stalen balken in het dak om de 5 meter en dit stramien is door de konstrukteur aangehouden. Omdat de hal zo groot was, is in het midden een dilatatie gemaakt. Aan het principe om om de 5 meter dakdragers te plaatsen werd niets afgedaan maar wel werden voor dit principe twee varianten ontworpen. In de eerste plaats werd een ligger bevestigd aan twee kolommen tot een portaal. De momenten die in de portaalkolommen zouden optreden door een belasting van een portaalkraan werden na berekening teniet gedaan door de portaalkraan op eigen kolommen te laten rusten (waarbij de portaalkraankolommen wel bevestigd werden aan de kolommen van de dakportalen om knik en kip te beperken). Na berekening bleek dat portalen in dit geval met een volle wandligger niet economisch zouden zijn: er werd veel te veel staal gebruikt. Bovendien zou door de gewenste vrije hoogte van 7 meter in de productiehal en de enorme hoogte van de vollewandligger meer buitenbeplating toegepast moeten worden (het gebouw wordt immers hoger). Er werden hierna op maat ontworpen vakwerkliggers gebruikt. Mocht de extra benodigde gevelbeplating duurder zijn geweest dan het kostenverschil tussen vollewandligger en vakwerkligger dan was gekozen voor de oplossing met vollewandligger. Voor de stabiliteit in dwarsrichting zijn de vakwerkliggers als portalen uitgevoerd. In de langsrichting zorgen windliggers in het dak langs de kopgevels en langs de dilatatievoeg voor stabiliteit. Omdat het gebouw een dilatatie kende in het midden en verder 130 meter lang was, was het niet zinvol het gebouw door een lange windligger stabiel te maken in de langsrichting. De windliggers zijn op de koppen van het gebouw geplaatst om te zorgen dat de afdracht van kip- en knikkrachten van de vakwerkliggers via trekstangen kan geschieden in de vorm van stalen strippen. Mochten de windliggers ergens van de kant af in het gebouwdak liggen dan moeten knik- en kipkrachten van de vakwerkliggers afgedragen worden door drukstaven naar de windliggers. Tussen de productiehal en de kantoorruimte was een "overgangsruimte" nodig en ontworpen die door de architect als een vierkant werd voorgesteld. Het probleem om een vierkant te overspannen werd hier vergemakkelijkt omdat de tussenruimte in staal mocht worden uitgevoerd en de konstruktie niet in het zicht behoefde te blijven hetgeen betekende dat via balken in één van de 2 richtingen (ondersteund met kolommen op circa 5 meter, een economische maat in staal) een geprofileerde staalplaat met damwandprofiel wordt

gedragen. De richting waarin overspannen wordt was dan verder afhankelijk waar kolommen geplaatst konden worden. Beide varianten wat betreft richting van overspannen werden bekeken en vervolgens werd de variant gekozen waar de meest praktische kolomplaatsingen bij voorkwamen. De stabiliteit van de vierkante tussenruimte werd geregeld door windliggers en windbokken in beide richtingen. Omdat tussen kantoor en tussenruimte enerzijds en tussen tussenruimte en productieruimte anderzijds brandwerende muren noodzakelijk waren, diende elk gebouw (kantoor, tussenruimte, hal) in zijn eigen stabiliteit te voorzien. Als het laatstgenoemde geval niet van toepassing zou zijn, zou het raadzaam zijn geweest het hoogste gebouw stabiel te maken, en de andere gebouwen hier op af te steunen en in geen geval het lagere gebouw stabiel te maken om hierop de hogere gebouwen af te steunen.

Vragen uit Bauwerk, Tragwerk, Tragstruktur

Is het, bij het zien van het bouwwerk, direct mogelijk de fysieke konstruktie te onderscheiden?

Ja, de stalen vakwerkspanten van de grote productiehal zijn te zien evenals de stalen geprofileerde dakplaten met damwandprofiel. Het dragend metselwerk bij het kantoor is eveneens zichtbaar in tegenstelling tot het betonnen dak. Bij de tussenruimte zijn wel kolommen zichtbaar (waarschijnlijk ingepakt) maar niet het dak op sommige plaatsen (systeemplafonds). Op deze plaatsen zijn dan ook geen balken aanwezig.

Is het, bij het zien van de fysieke konstruktie, direct mogelijk een schematisatie van de konstruktie te maken?

Bij de productiehal is dit zeer goed mogelijk al moet opgemerkt worden dat de kraanbaan enige verwarring kan scheppen. Omdat kolommen en liggers van staal zijn is het goed mogelijk te schematiseren. Bij het kantoor is, bij het zien van de ruwbouw, duidelijk te zien hoe de vloeren overspannen en waar ze opliggen.

Is de vertaling van de fysieke draagkonstruktie naar een schematisatie slechts door grote idealiseringen mogelijk?

Nee.

Is de fysieke draagkonstruktie niet op meerdere manieren te schematiseren?

Nee, met uitzondering van het feit dat op samenwerking van de voorspanvloeren is gerekend bij het verzorgen van de stabiliteit in de dwarsrichting: de vraag is niet of dit wel mag maar of dit ook fysiek werkt.

Zorgt de gekozen draagstructuur voor de gunstigste draagsituatie in het bouwwerk?

Tijdens de procesbespreking van dit project is duidelijk geworden dat hier veel aandacht aan is besteed.

Is de fysieke draagkonstruktie, met de bouwmethode in beschouwing genomen, het meest doelmatig ontworpen met de draagstructuur als achterliggend schema?

Het is niet bekend in hoeverre het toepassen van voorgespannen holle kanaalplaatvloeren de meest gunstige lastenverdeling geeft omdat deze vloer is gekozen omwille van uitvoeringstechnische redenen.

Sluit de gekozen konstruktie aan bij de stijl van het gebouw?

Waar de konstruktie zichtbaar is (hal en tussenruimte) is deze vanzelfsprekend aanwezig. In het kantoor wordt de konstruktie hoogstwaarschijnlijk weggewerkt.

3.4.4 Nieuwbouw Eindhovense instelling voor Gezinsverzorging/Thuishulp

Gebouw: Gezinsverzorging/Thuishulp Eindhoven
Architect: OD 205, T.P. van der Zanden hbo, architekt BNA
Konstrukteur: Advies- en Ingenieursburo van de Laar bv

Globale planverkenning en globale konstruktieverkenning

Een nieuwbouw voor een instelling voor Gezinsverzorging/Thuishulp dient om de administratieve werkzaamheden te herbergen en zogenaamde managementteams, bestaande uit ongeveer 8 mensen, die leiding geven aan de mensen die daadwerkelijk huishoudelijke hulp verlenen. Zoals op de figuren 12, 13 en 14 te zien is bestaat het gebouw uit drie lagen waarbij de derde laag smaller is dan de twee onderliggende lagen. Op de onderste laag liggen een ontvangsthal, vergadercentrum, administratie, en een gemeenschappelijke ruimte. Op de eerste verdieping liggen kamers voor de leidinggevende teams en op de tweede verdieping liggen de kamers voor het management. De kopgevels van het gebouw zijn gesloten waarbij de langsgevels bestaan

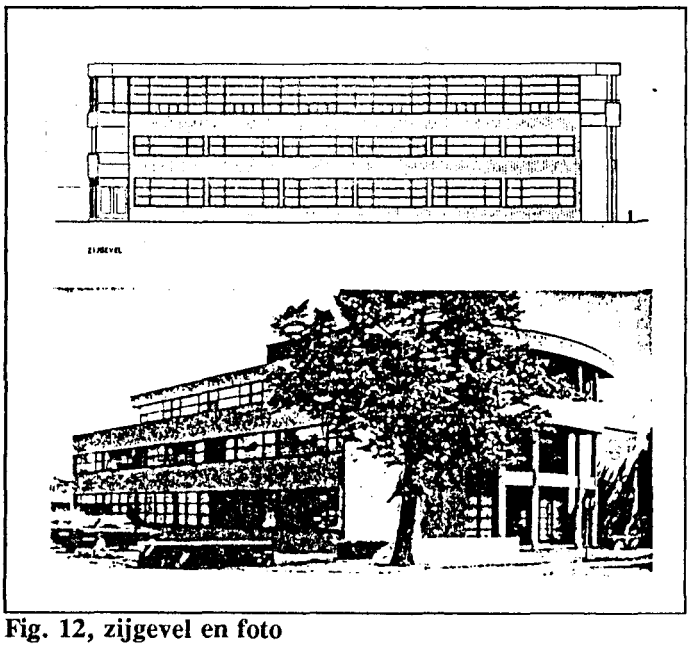


Fig. 12, zijgevel en foto

uit balkons, erkers en glas. De dakrand is aan de kopse kanten halfrond. Het betonnen skelet dat draagt is zichtbaar in het gebouw. De derde laag wordt gedragen door liggers in de dwarsrichting van de strook van de derde laag, waarbij de kolommen iets naar binnen geplaatst zijn. De balken van geprefabriceerd beton zijn in hoogte aangepast aan het momentenverloop in de balken. Omdat de eerste en tweede laag hetzelfde balkensysteem kennen als de derde laag, maar veel breder zijn, liggen hier balken op, op de uiteinden van de eerst genoemde balken en met de andere hoek op de aldus dragende wanden.

Procesgang

In het begin was het de bedoeling de Gezinsverzorging/Thuishulp Eindhoven te huisvesten in het oude St. Jorishuis. Toen dit huis in 1987 afbrandde werd besloten een nieuwbouw te ontwerpen voor de huisvesting van de Gezinsverzorging Eindhoven. Een programma van eisen werd gemaakt waarin te gebruiken ruimten met benodigde oppervlakte per ruimte en gewenste laag waarop de ruimte lag vermeld waren. Het gehele gedeelte ruimten bestemd voor het management moest op de derde laag liggen. De leiding van de huishoudelijke hulpen moest plaats kunnen nemen op de eerste laag en specifieke ruimten als een telefooncentrale, huishoudelijke dienst en dergelijke werden geacht op de begane grond te liggen. Mede door de situatie die volgens de architect vroeg om een naar boven verjongend gebouw werden de eerste twee lagen als een soort standaard kantoorlagen ontworpen waarbij de derde laag smaller was dan de twee onderliggende lagen en inderdaad het management van de stichting huisvestte. In een vroeg stadium van het ruimtelijk ontwerp werd reeds gesproken met het adviesburo van de Laar bv. Uitgangspunten waren dat er een bijzonder gebouw moest komen, dat wil zeggen, dat standaardkonstrukties niet erg wenselijk waren. De middenzone van het gebouw moest open blijven omdat deze zone een verkeerszone en een openbare zone van het gebouw was. En verder moest de bouwtijd erg kort zijn omdat door het afbranden van het St.-Jorishuis reeds beschikbare tijd verdwenen was. Een aantal konstruktievarianten werd bekeken en ditmaal werd een variant gekozen die bijzonder was hetgeen gewenst was bij het programma van eisen. Omdat vanuit het ruimtelijk ontwerp reeds besloten was, grote lichtkappen toe te passen, en veel sparingen in de vloeren aangebracht moesten worden, werd besloten

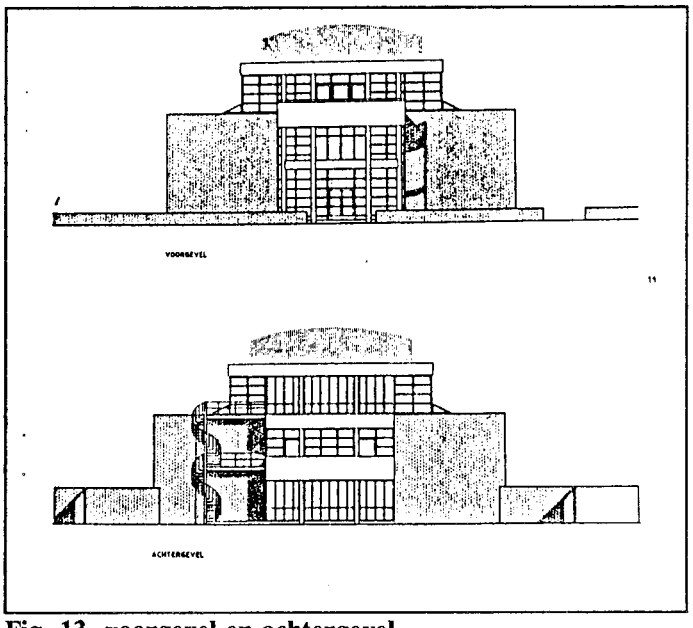


Fig. 13, voorgevel en achtergevel

om breedplaatvloeren toe te passen die relatief eenvoudig toelaten sparingen toe te passen. Door een dwarsbalkensysteem toe te passen werd bereikt dat de middenzone open bleef. Omdat ook de gevel nogal wat variatie kende kwamen onder de dwarsbalken alleen ter plaatse van de scheiding tussen kantoor- en openbare zone kolommen. Met een computerprogramma werd de balk precies zo gemaakt dat de doorsnede van de balk telkens aange-

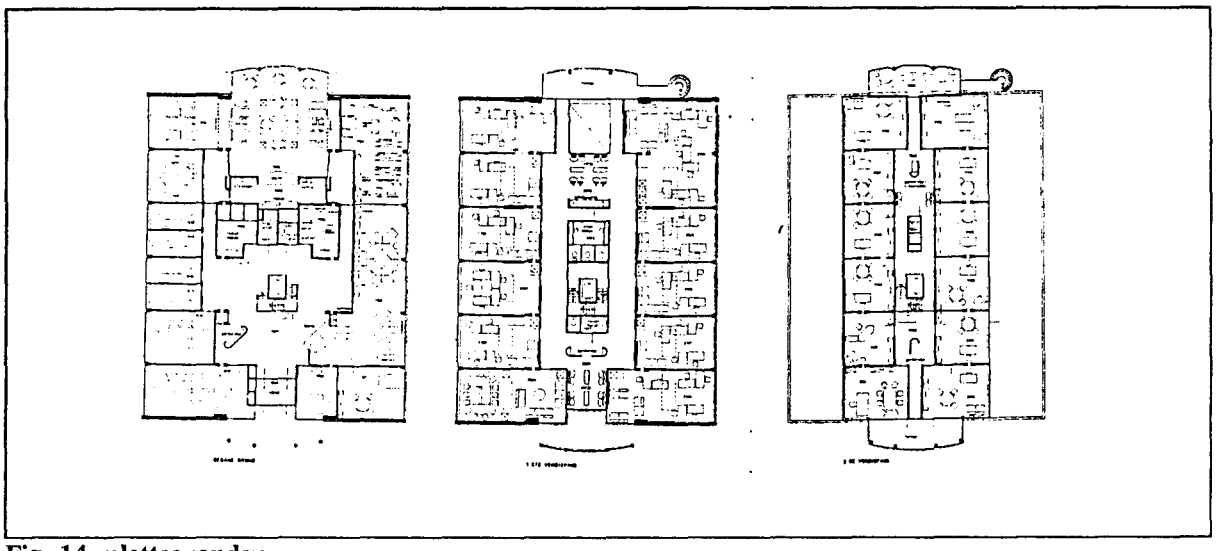


Fig. 14, plattegronden

past was aan de inwendige momenten ter plaatse. Aan de uiteinden van de randbalken zorgen prefab borstweringsplaten voor de gevel. Omdat de betonnen konstruktie in het zicht zou blijven werd elk detail zeer goed bekeken, niet alleen om het detail snel uitvoerbaar te maken, maar ook om het detail er esthetisch fraai uit te laten zien. De kolommen onder de fraaie dwarsbalken zijn 300 * 1200 mm., zodoende is de stabiliteit in dwarsrichting goed verzorgd. In de langsrichting werd voor stabiliteit gezorgd door gewapende binnenwanden in de langsriching. De dwarsbalken, die op 5,40 meter h.o.h. lagen, bepaalden meteen de kamermaat. Al is op de begane grond het stramien van 5,40 meter enkele keren opgedeeld in kleinere ruimten.

Vragen uit Bauwerk, Tragwerk, Tragstruktur

Is het, bij het zien van het bouwwerk, direct mogelijk de fysieke konstruktie te onderscheiden?

Ja, het gehele dwarsbalkensysteem is zichtbaar en ook de stabiliteitswanden zijn zichtbaar.

Is het, bij het zien van de fysieke konstruktie, direct mogelijk een schematisatie van de konstruktie te maken?

Ja, het is duidelijk dat de kolom-dwarsbalk-verbinding momentvast is en de ligger zelf spreekt voor zich.

Is de vertaling van de fysieke draagkonstruktie naar een schematisatie slechts door grote idealiseringen mogelijk?

Nee.

Is de fysieke draagkonstruktie niet op meerdere manieren te schematiseren?

Onmogelijk omdat de gekozen detailleringen en materialen eenduidig te schematiseren zijn.

Zorgt de gekozen draagstructuur voor de gunstigste draagsituatie in het bouwwerk?

De dwarsbalken zelf zijn erg efficient omdat de doorsnede hoger wordt daar waar ook de momenten in de balk hoger worden. De doorsnede volgt in hoogte precies de momentenlijn van de balk. De hoofddraagstruktuur had echter wellicht veel efficienter kunnen zijn en het adviesbureau ontkent dit ook niet: het belang van een esthetische konstruktie (op zichzelf efficient) was groter dan een goedkope konstruktie. Een systeem met gewone liggers en kolommen was wellicht goedkoper geweest maar niet konstruktief efficienter!

Is de fysieke draagkonstruktie, met de bouwmethode in beschouwing genomen, het meest doelmatig ontworpen met de draagstructuur als achterliggend schema?

Ja, als de schematisatie als onveranderlijk gegeven beschouwd wordt, dan kan in dit geval de fysieke draagkonstruktie niet beter.

Sluit de gekozen konstruktie aan bij de stijl van het gebouw?

Er werd een bijzonder konstruktie gevraagd en bovendien moest de middenzone van het gebouw vrijblijven. Hoewel de oplossing van de eerste eis altijd een subjectieve is, zijn beide eisen vervuld. De konstruktie blijft in het zicht en er mag dus aangenomen worden dat de konstruktie past bij de stijl van het gebouw.

3.5 Logica/kennissystemen/kennisrepresentatie/interne opbouw computer

In de twee aan dit afstudeerproject voorafgaande projecten is uitvoerig aandacht besteed aan verdieping van de kennis omtrent logica, kennissystemen, kennisrepresentatie en de interne opbouw van een computer. De verslagen van deze projecten zijn als bijlage bij dit afstudeerproject beschikbaar, de bijlage "T8-T9-verslagen". De volgende colleges aan de TUE zijn gevolgd:

2L100 Logica 1
2L110 Logica 2
2L340 Kennissystemen
5A030 Inleiding computers

4 Kennis omzetten in processen en data

4.1 Inleiding

Alle opgenomen en verwerkte kennis omtrent konstruktief ontwerpen en de achterliggende struktuur van computergebruik kan nu omgezet worden in processen en data. Om deze processen en data te beschrijven wordt een procesmodel IDEF0 en een datamodel IDEF1x gebruikt. Beide modellen worden gebruikt in het Bouw-Informatie-Model (BIM) [3]. Er volgt eerst een korte kennismaking met het BIM.

4.2 BIM Bouw Informatie Model

Een bouwinformatiemodel geeft verkeersregels voor de uitwisseling van informatie die van belang is bij een bouwproces. Dit bouwproces is niet alleen het feitelijk materieel realiseren van een bouwwerk maar ook, en in het bijzonder, het ontwerpen, het konstrueren, en het beheren van een bouwwerk. Het bouwinformatiemodel BIM is bedoeld als referentiepunt voor alle vragen die zouden kunnen onstaan tijdens het informatiseren van een bouwproces. Het BIM is een familielid van de bekende produktmodellen en modelleert als zodanig ook het bouwproces. Het bouwinformatiemodel is gesplitst in een procesmodel en een gegevensmodel. In het procesmodel worden de processen die een rol spelen bij het bouwproces gemodelleerd. In het gegevensmodel wordt struktuur aangebracht in de gegevens die in een bouwproces een rol spelen. Erg interessant is dat het BIM een metamodel bevat. Zoals in de logica de metalogica gebruikt kan worden om te bewijzen dat de syntactische systemen soms volledig en correct kunnen zijn, zo wordt met het metamodel het BIM beschreven.

Het procesmodel van het BIM wordt op een bepaalde wijze weergegeven door de methode met de naam IDEF0. Deze methode maakt afspraken over hoe processen kunnen worden geklassificeerd en met elkaar in verband kunnen worden gebracht door gegevensstromen tussen deze processen. Een proces wordt weergegeven door een rechthoek alwaar informatie binnenkomt aan de linkerzijde en informatie naar buitengaat aan de rechterzijde. Deze informatie kan ook opgevat worden als energie, materiaal en mankracht wanneer het proces een materiaalbewerking voorstelt.

Het gegevensmodel wordt met de methode IDEF1x weergegeven. Rechthoeken geven entiteiten aan. Een entiteit wordt gedefinieerd als iets dat een significante betekenis heeft voor de gebruiker en waarvoor het wenselijk is gegevens vast te leggen. Deze entiteiten kunnen eigenschappen bezitten (zo heeft de entiteit kozijn eigenschappen zoals merk, fabricagedatum, hoogte, enz.). Deze eigenschappen vormen nieuwe entiteiten en rechthoeken. De rechthoeken kunnen met elkaar verbonden worden opdat relaties tussen entiteiten mogelijk zijn. Om een entiteit eenduidig te kunnen noemen heeft een entiteit minstens 1 sleutel: een codering of naam voor de entiteit (kozijn A 10-12 bijvoorbeeld).

In het bouwinformatiemodel is het van belang dat afgezien van de proces- en gegevensmodellen ook het gebouw zelf (en niet het bouwproject) goed gerepresenteerd wordt. Het gebouwmodel is een gegevensmodel en bestaat uit een ruimtelijk systeem en een technisch systeem. Het ruimtelijk systeem bestaat uit ruimtedelen die opgebouwd zijn uit ruimten. Het technisch systeem bestaat uit gebouwdelen die opgebouwd zijn uit componenten. De relatie tussen de ruimten en de componenten wordt vastgelegd met behulp van de topologie.

Het model wordt bezien vanuit de mogelijkheden die het kan bieden bij dit afstudeerproject. De nummering van de processen is eenvoudig: wanneer een nummer wordt bijgevoegd zonder dat de voorgaande nummers gewijzigd worden is er sprake van een proces dat, als onderdeel van het bovenliggend proces, een bovenliggend proces beschrijft. Het procesmodel is dan als volgt te volgen:

A-1 Omgeving Bouwproject
1 Houdt bouwregels in stand
2 Ontwikkel bouwproject
AO Ontwikkel bouwproject
A1 Bestuur bouwproject
A2 Voer bouwproject uit
A21 Formuleer object definitie
A22 Maak object specificatie
A221 Ontwerp object specificatie
A2211 Ontwerp bouwkundig subsysteem
A22111 Maak ruimtelijk ontwerp
A22112 Maak materieel ontwerp
A22113 Maak detailontwerp
A2212 Ontwerp draagconstructie
A22121 Ontwerp funderingsconstructie
A22122 Ontwerp constructief hoofdsysteem
A22123 Dimensioneer constructie
A222 Controleer prestatie
A223 Maak object representatie
A224 Stel realisatie eisen vast
A23 Realiseer fysiek object
A24 Gebruik fysiek object

De onderlijnde processen zijn van belang in dit afstudeeronderzoek. Deze processen worden niet uitgewerkt in het BIM. Een beschrijving van de processen is wel aanwezig.

A21 Formuleer object definitie

Het vastleggen van eisen omtrent het te maken ontwerp, waarbij nog geen specificaties hoeven te worden vastgelegd. De invoer van een programma zou als deel van het proces A21 kunnen worden beschouwd (Programma van Eisen of objectdefinitie).

A22111 Maak ruimtelijk ontwerp

Het ontwerpen van de ruimten en de objectdelen qua plaats en maat

A22112 Maak materieel ontwerp

Het ontwerpen van de objectdelen qua dimensionering en samenstelling van de materialen

A22122 Ontwerp constructief hoofdsysteem

Het ontwerpen van het constructief hoofdsysteem omvat o.a. het bepalen van het constructietype, de constructiematerialen en de hoofdafmetingen.

In het BIM datamodel ("datamodel" is gelijk aan "gegevensmodel") zijn de gegevensstructuren van belang die de opbouw van een bouwproject beschrijven. Het gebouw is te splitsen in een ruimtelijk systeem en/of in een technisch systeem. Het verband tussen de ruimten en de technische componenten is de topologie. Er zijn een aantal gegevensmodellen. Deze gegevensmodellen hoeven echter niet in elkaar genest te zijn dat wil zeggen dat een gegevensmodel een uitgewerkte versie van een entiteit van een ander gegevensmodel kan zijn. Hieronder worden de van belang zijnde gegevensmodellen kort besproken:

D10 Gegevensmodel objectsamenstelling

Het object wordt onderverdeeld in delen die in subdelen worden onderverdeeld. Er wordt doorgegaan met verdelen totdat het te beschouwen voorwerp of onderdeel is bereikt.

D11 Gegevensmodel de samenstelling van de eenheid

Het te beschouwen voorwerp of onderdeel kan in delen worden verdeeld die vervolgens in subdelen kunnen worden verdeeld.

D12 Gegevensmodel topologiemodel

In dit model wordt het duidelijk dat delen van het gebouw weergegeven worden door een punt, lijn, vlak of ruimte. Deze weer te geven delen kunnen zowel technische componenten zijn alsmede ruimten.

D13 Gegevensmodel de topologische elementen 1 / D14 Gegevensmodel de topologische elementen 2

Tussen ruimte, vlak, lijn, en punt zijn diverse relaties mogelijk. Een punt kan het begin zijn van een lijn, een ruimte kan in een andere ruimte liggen, enz. Deze gegevensmodellen behandelen deze relaties.

D15 Gegevensmodel subsamenstelling van het object en de topologie

Een objectsubsamenstelling zoals genoemd in gegevensmodel D11 kan worden gekoppeld aan de gegevensmodellen D13 en D14. Ieder subonderdeel krijgt een dimensie en een plaats.

Het detailmodel Constructief ontwerp van het BIM bevat de decompositie van de processen Ontwerp constructief hoofdsysteem en Controleer draagconstructie. In het detailmodel Constructief ontwerp is voor dit afstudeerproject het ontwerpen van de draagconstructie van belang. Het detailleren, controleren, schematiseren en beoordelen van de constructie is niet van belang. Het ontwerpproces wordt in het BIM behandeld als volgt:

Ontwerpen constructie omvat o.a. het bepalen van konstruktief hoofdsysteem, het constructietype, de constructiematerialen en de hoofdafmetingen. In de gegevensmodellen constructie wordt een konstruktie beschreven als resultaat van de processen Ontwerpen constructie en Detailleren constructie. Omdat veel Engelse woorden gebruikt worden is een lijst met vertalingen bijgevoegd. Een Structure wordt opgedeeld in een aantal Assemblies die opgedeeld worden in Parts. Parts worden verbonden met Part_joints. Features zijn eigenschappen van een deel van een Part, bijvoorbeeld een deeloppervlaktebewerking.

Het topologiemodel van het gegevensmodel constructie is niet het topologiemodel van het kernmodel van het BIM. Wel zijn er verbindingen tussen deze twee modellen. Het topologiemodel van het gegevensmodel constructie omvat Nodes en Structural Links. Parts worden beschreven door Procedural_shape_definition, Csg_representation of Boundary_representation. In het BIM is het F.E.M. aanwezig: een model dat de gegevensstroom van eindige-elementen-berekeningen behandeld.

Structure	Constructie
Assembly	Samenstel van parts
Parts	Onderdelen
Part_joints	Verbindingen van onderdelen
Feature	Optie, eigenschap van deelgebied van een onderdeel
Nodes	Knooppunten
Structural Links	Verbindingen of delen tussen knooppunten

Procedural_shape_definition	Een beschrijving van een vorm die vertelt wat de vorm doet en hoe de vorm functioneert bij de functievervulling van een onderdeel
Csg_representation	C.S.G.-representatie van een object, zie voor nadere informatie omtrent C.S.G. de bijlage "T8-T9-verslagen".
Boundary_representation	Boundary-representatie

4.3 Procesmodel konstruktief ontwerpen

Nu het proces konstruktief ontwerpen wat nader bekeken is, zowel in de praktijk als door middel van literatuuronderzoek, kan dit proces voorzichtig op een wat hoger niveau bekeken worden. Na het ruimtelijk ontwerp (1), dat minimaal aan de gestelde eisen (het Programma van Eisen PvE) moet voldoen en verder vaak een niet functionele verdere uitwerking kent, worden een aantal konstruktieve hoofddraagvarianten(2) voorgesteld door

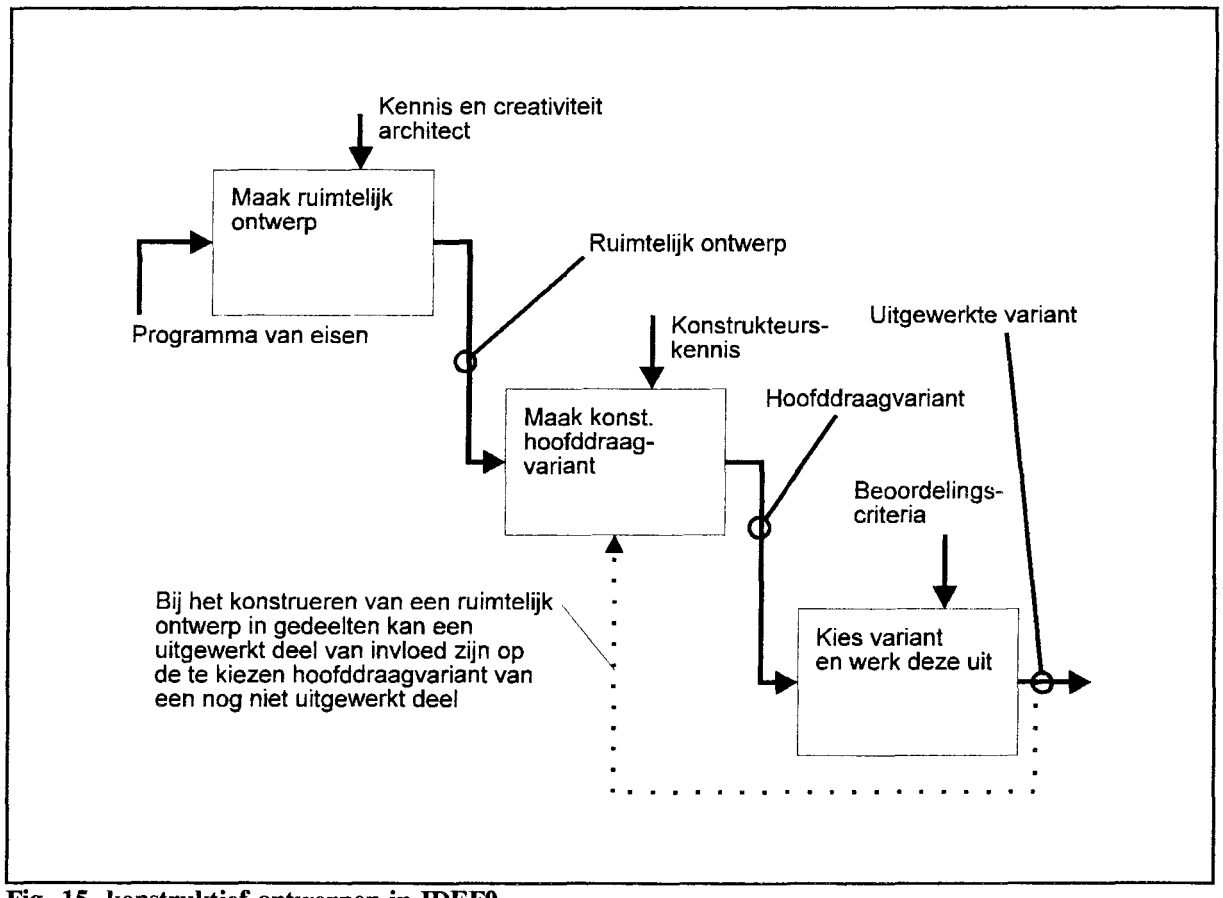


Fig. 15, konstruktief ontwerpen in IDEF0

de konstrukteur, die ook weer, net als bij het ruimtelijk ontwerp, minimaal moeten doen

wat in de eisen gesteld wordt. Vervolgens wordt uit de hoeveelheid konstruktieve hoofddraagvarianten, een variant gekozen(3) die hetzij minimaal aan de gestelde voorwaarden voldoet en economisch is, hetzij gekozen wordt omdat de andere varianten een belangrijk facet van het ruimtelijk ontwerp zoals de ontwerper dat wil onmogelijk maken. Samengevat gebeurt er voor de konstrukteur het volgende:

1) Er is een programma van eisen

2) Er wordt een ruimtelijk ontwerp gemaakt door de architect

2) Er worden een aantal konstruktieve hoofddraagvarianten bedacht door de konstrukteur

3) Een van deze varianten wordt tijdens gezamelijk overleg tussen architect en konstrukteur gekozen, vaak om economische redenen, maar zonder wezenlijke beperkingen op te leggen aan het ruimtelijk ontwerp

Dit proces kunnen we ook in IDEF0, een kennisrepresentatietechniek, vertalen. IDEF0 is onder andere toegepast in het BIM, het bouwinformatiemodel. In figuur 15 is dit procesmodel te zien.

4.4 Omzetting kennis, functionele eisen, uitvoering

Voordat de kennis omgezet wordt in processen en data moet gekeken worden welke elementen van het generiek konstruktieproces nu van belang zijn om mee te nemen op basis van de theorie gevonden in de literatuur en de behandelde vier cases. Met nadruk wordt er op gewezen dat de zaken die van belang worden gevonden voor een ieder verschillend kunnen zijn. Zo zal ook voor een ieder een ander samenstelsel van processen en data ontstaan.

De gekozen werkwijze om zowel de kennis om te zetten in processen en data als de processen en data om te zetten in algoritmen is gebaseerd op het ontwikkelen van een prototype (het programma). Een zoekende, exploratieve ontwikkeling. De resultaten hiervan zijn, deels informeel, voor de bouwkundige/konstrukteur begrijpbaar beschreven in 5.2 tot en met 5.7.

Elementen van het generiek konstruktieproces die van belang worden gevonden:

1) Ten eerste is het proces van belang zoals dat te zien is in figuur 15. Dit proces is aanwezig in alle cases en een vereenvoudiging van het proces dat in de literatuur beschreven is, te zien in de tweede afbeelding van figuur 1. Dit proces bevat impliciet veel informatie en handelingen.

2) In de literatuur is te lezen dat een van de problemen bij het omzetten van een konstruktief ontwerpproces in een algoritme is, dat met de scheppende rol van de mens in het ontwerpproces geen rekening gehouden wordt.

3) Het blijkt dat gekonstrueerd wordt met bepaalde klassen en verzamelingen konstruktie-elementen, te zien in de tweede afbeelding van figuur 2 en de eerste afbeelding van figuur 3.

4) Het blijkt dat de konstruktie-elementen veelvuldig in bepaalde geordende groepen voorkomen om een ruimte te overspannen, zie bijvoorbeeld de tweede afbeelding van figuur 4.

5) Het ruimtelijk ontwerp is een moeilijk vast te leggen proces en het resultaat wordt de konstrukteur meestal aangeboden.

6) In de cases is echter duidelijk dat een ruimtelijke vorm kan veranderen als de konstruktiekeuze daartoe aanleiding geeft. Een ruimte wordt bijvoorbeeld verplaatst. De ruimtelijk vorm geeft ook een voorwaarde voor de konstruktiekeuze.

7) Uit de literatuur is bekend dat niet hulpmiddelen ter systematisatie gebruikt moeten worden die reeds succesvol zijn bij de analyse van ontwerpen (bijvoorbeeld rekenapparatuur). Zie voor verder informatie over dit punt 3.3.

8) De mens gebruikt combinaties van kwalitatieve eisen bij het ontwerpen.

Uit de antwoorden op de vragen van Bauwerk, Tragwerk, Tragstruktur bij de vier cases zijn ook belangrijke elementen van het konstruktief ontwerpproces af te leiden. Belangrijk is dat slechts vier willekeurige cases zijn bekeken, zodoende is het antwoord nooit volledig representatief. Bij elke vraag wordt kort het commentaar vermeldt.

Vraag: Is het, bij het zien van het bouwwerk direct mogelijk de fysieke konstruktie te onderscheiden?

Tweemaal het antwoord "ja", eenmaal wordt de konstruktie ingepakt in wat afwerkingsmateriaal.

Vraag: Is het, bij het zien van de fysieke konstruktie, direct mogelijk een schematisatie van de konstruktie te maken?

Driemaal een bevestigend antwoord.

Vraag: Is de vertaling van de fysieke draagkonstruktie naar een schematisatie slechts door grote idealiseringen mogelijk?

Driemaal een ontkennend antwoord. Tezamen met het antwoord op de vorige vraag volgt hieruit een belangrijk element voor het konstruktief ontwerpproces:

9) Zoals geschematiseerd wordt, wordt ook vaak gekonstrueerd. Het wil echter niet zeggen dat zoals geschematiseerd wordt, ook gekonstrueerd moet worden.

Vraag: Is de fysieke draagkonstruktie niet op meerdere manieren te schematiseren?

In twee van de vier gevallen is niet geheel duidelijk hoe de konstruktie zich nu werkelijk zal gedragen. Er zijn aannamen gedaan. Hieruit volgt een element van het konstruktief ontwerpproces:

10) Er worden aannamen gedaan over het gedrag van konstrukties die tot gevolg hebben dat de ogenschijnlijke schematisatie van de konstruktie niet correct hoeft te zijn.

Vraag: Zorgt de gekozen draagstructuur voor de gunstigste draagsituatie in het bouwwerk?

Bij een case blijkt dit moeilijk te beoordelen, de konstrukteur beweert van wel. Bij de volgende case beweert de konstrukteur eveneens hieraan te hebben voldaan en lijkt hij gelijk te hebben. Bij de vierde case is bewust voor een esthetisch fraaie konstruktie gekozen die niet aan de vraag beantwoord. Het volgende element van het konstruktief ontwerpproces is van belang:

11) In een konstruktief ontwerp-proces wordt of een poging gedaan de gunstigste draagsituatie van een konstruktie te verkrijgen oftwel aan esthetische eisen de voorkeur gegeven (of beide).

Vraag: Is de fysieke draagkonstruktie, met de bouwmethode in beschouwing genomen, het meest doelmatig ontworpen met de draagstructuur als achterliggend schema?

De antwoorden op deze vraag zijn niet van belang omdat in dit afstudeerproject de bouwmethode niet meegenomen wordt als onderdeel van de ontwerpvoorwaarden.

Vraag: Sluit de gekozen konstruktie aan bij de stijl van het gebouw?

Bij de eerste case sluit de gekozen konstruktie beslist niet aan bij de stijl van het gebouw. Bij de andere cases sluit de gekozen konstruktie duidelijk wel aan bij de stijl van het gebouw, waarbij bij de laatste case de konstruktie duidelijk een bijdrage levert aan de stijl van het gebouw.

Hierdoor ontstaan de volgende functionele eisen voor het programma:

- Op basis van een beperkte invoer van een aantal ruimten en de afmetingen hiervan worden een aantal konstruktief ontwerpen gepresenteerd, die globaal gedimensioneerd worden. Deze eis is een logisch gevolg van de opzet van dit afstudeerproject om een programma te maken dat het konstruktief ontwerpproces ondersteunt.

- De gebruiker moet tijdens het genereren van deze konstruktief ontwerpen kunnen ingrijpen en veranderingen mogelijk maken. Deze eis volgt uit het voorgaande punt 2.

Dit programma bestrijkt het gehele proces konstruktief ontwerpen, maar bereikt bij elk aspect van het proces slechts een geringe diepgang:

- De omzet van de invoer (ruimten, afmetingen en voorwaarden) tot een ruimtelijk ontwerp. Deze eis vloeit voort uit het voorgaande punt 5. Om een konstruktief ontwerp te maken is een ruimtelijk ontwerp nodig. Deze wordt gegenereerd aan de hand van deze eis.

- De konstruktie van het ruimtelijk ontwerp en globale dimensionering met als beperking een beperkt aantal konstruktie-elementen en konstruktievarianten. Deze eis ontstaat uit het voorgaande punt 3 en 4.

- De effecten van ruimtelijk ontwerp en konstruktief ontwerp op elkaar, en de terugkoppeling naar vorige ontwerpstappen, komt van het voorgaande punt 6.

- Evaluatie bij elke ontwerpstap en aanpassing ontwerp. Een logisch gevolg van het proces bij het voorgaande punt 1.

Het voorgaande punt 8 waarbij vertelt wordt dat mensen combinaties van eisen kunnen verwerken is niet opgenomen in de functionele eisen voor het programma. In punt 7 wordt vertelt dat geen conventionele wegen bewandeld moeten worden bij het zoeken naar hulpmiddelen om het proces konstruktief ontwerpen te ondersteunen. Daarom is de conventionele procedurele aanpak met de nodige voorzichtigheid bekeken. De punten 9 t/m 11 zijn moeilijk te verwerken in functionele eisen en er wordt volstaan met het noemen van deze elementen van het konstruktief ontwerpproces.

In het Bouw Informatie Model zijn, zoals reeds behandeld drie wezenlijk belangrijke processen aan te wijzen voor dit afstudeerproject. Dat zijn:

Maak ruimtelijk ontwerp
Maak materieel ontwerp
Ontwerp constructief hoofdsysteem

Deze processen staan met elkaar in verband, zoals te zien in het Bouw Informatie Model. Een datamodel, dat de data en datastromen beschrijft, geeft aspecten van de verbanden tussen de drie processen weer. Rekenfuncties bewerken en transformeren de gegevens in het datamodel.

Nu wordt een benadering voor deze samenhang tussen de drie processen gekozen. De benadering is beperkt, exemplarisch en kan verbeterd en uitgebreid worden. Andere benaderingen zijn mogelijk.

Definities:

- Een gebouw bestaat uit lagen (verdiepingen).
- Elke laag is overal even hoog en sluit aan op een eventueel onder- of bovenliggende laag.
- Een gebouw kent geen fundering: onder het gebouw ligt een oneindig stijve plaat.
- In een laag bevinden zich ruimten.
- Ruimten zijn rechthoekig, hebben een breedte en lengte, en hebben de hoogte van de laag.
- Er zijn voorwaarden die vertellen dat ruimten op een laag bij elkaar dienen te liggen.
- Clusteringen van ruimten op een laag die een rechthoekige plattegrond hebben, waarbij de gehele omtrek van de rand van de plattegrond bestaat uit randen van ruimten, zijn zones.

- Konstruktie-elementen zijn:

  Houten balk/kolom
  Stalen kolom
  Stalen balk
  Betonnen kolom
  Betonnen schijf
  Stalen vloerplaatelement
  Betonnen vloerplaatelement
  Stalen ruimtevakwerkelement

- Konstruktievarianten voor zone's zijn:

  Elementen houten balk/kolom gevormd tot portalen die tezamen de zone bestrijken. Dit is een niet vlakke dakvorm, alleen bruikbaar bij ruimten in de bovenste laag.

  Stalen kolomen met balken tot raamwerk gevormd, waarop stalen vloerplaatelementen liggen.

Betonnen schijven waarop betonnen vloerplaatelementen liggen.

Betonnen kolommen op de hoeken van een ruimtevakwerk.

Akties:

De ruimten worden in de laag, per laag beginnende bij de eerste, geplaatst, volgens de voorwaarden. Is reeds een voorgaande laag met konstruktie aanwezig, dan wordt het ruimtelijk ontwerp hierop afgestemd. Alle mogelijkheden worden gemaakt.

In de laag worden zones ontwikkeld. Alle mogelijkheden worden gemaakt uit alle mogelijkheden van de voorgaande aktie.

De zones in de laag worden gekonstrueerd. Alle mogelijkheden worden gemaakt, maar sommige mogelijkheden uit de volgende aktie zullen niet meer gebruikt kunnen worden.

De ruimten van de volgende laag worden geplaatst, de volgende laag wordt gezoneerd en gekonstrueerd, enz.

Deze akties zijn niet geheel "generate & test" (zie de bijlage "T8-T9-verslagen") maar de gegenereerde ontwerpen zijn al ten dele afgestemd op de voorwaarden van de invoer en de beperkingen van de konstruktie-elementen (feasibility) door de wijze van genereren.

Over deze akties heen loopt een controle van de gebruiker die bepaalde oplossingen of oplossingsgroepen verwijderd: een "test"-onderdeel. Andere "test"-technieken (niet toegepast in deze benadering) zouden kunnen zijn:

- De samenhang van de konstruktie-onderdelen over verschillende lagen.
- De uitvoerbaarheid van de konstruktie.
- Een van boven naar beneden lopende controleberekening van het ontwerp.

Een nadere specificatie van de akties is te lezen in hoofdstuk 5.

De redenen waarom voor onderstaande benadering is gekozen zijn:

- De benadering is te combineren met bestaande technieken voor space-allocation, hoewel de benadering een eigen eenvoudige techniek van space-allocation gebruikt.
- De "zonering" lijkt een goede methode om konstruktie-mogelijkheden te zien. Het lijkt een vervanging van de intuitie van de konstrukteur om bepaalde clusters van ruimten te konstrueren.
- De konstruktie van de zones werkt met shape-grammars: bij een bepaalde vorm van een zone horen bepaalde konstruktiemogelijkheden.

Door de gekozen, nog onvolledige en exemplarische benadering zijn de volgende beperkingen aanwezig:

- Ruimten zijn altijd rechthoekig.
- Er wordt geen rekening gehouden met de fundering.
- Er wordt geen rekening gehouden met doorgangen tussen ruimten en openingen zoals ramen.
- Alleen op de laatste laag zijn niet-vlakke dakvormen mogelijk, terwijl ook ruimten op lagere lagen die boven vrij liggen deze dakvormen zouden moeten kunnen gebruiken.
- Het ontwerpen gaat per laag.

Bij de nu aanwezige programma-uitwerking zijn de volgende beperkingen, die niets met de gekozen benadering te maken hebben, van belang:

- Afmetingen van ruimten zijn altijd even.
- De enige voorwaarde tussen ruimten is de voorwaarde die vertelt dat twee ruimten langs elkaar moeten liggen.
- Er zijn geen konstruktie-onderdelen in de ruimte aanwezig, slechts langs de ruimtelijke referentie.
- Door een hybride (procedurele-declaratieve) programmering is het niet zo snel mogelijk om bijvoorbeeld de konstruktie van een onderliggende laag te laten afhangen van een konstruktie van de bovenliggende laag, met andere woorden erg flexibel met de data om te springen.
- Er zijn een beperkt aantal konstruktie-elementen en globale konstruktievarianten.
- Bij veel ruimten en lagen treedt een lange verwerkingstijd op.
- Er wordt bij het konstrueren geen rekening gehouden met stabiliteit.
- Er wordt bij het dimensioneren van een laag geen rekening gehouden met de bovenliggende ruimten en konstruktie-elementen. Bij de verdeling van ontwerpen in de groepen hoogbouw, laagbouw, en verdiepingsbouw van circa 3 tot 6 lagen, hoeft dit echter niet zo'n groot probleem te zijn.

Worden deze eisen in mogelijke processen en data vertaald dan kunnen deze beschreven worden in proces- en datamodellen die ook in het BIM gebruikt worden.

4.5 Deel algoritme in BIM

In het BIM wordt het konstruktief ontwerpproces gevonden in de modellen A22111 Maak ruimtelijk ontwerp en A22122 Ontwerp constructief hoofdsysteem.

4.5.1 Procesmodel Konstruktief Ontwerpen

Een bedacht algoritme voor het konstruktief ontwerpproces is in dit afstudeerproject moeilijk te vertalen in een BIM-model omdat per laag van een gebouw ontworpen wordt, eerst ruimtelijk, vervolgens konstruktief. Toch is geprobeerd A22111 Maak ruimtelijk ontwerp en A22122 Ontwerp konstruktief hoofdsysteem uit te werken. Dit is te zien in figuur 17, A2 Maak ruimtelijk ontwerp en figuur 18, A3 Ontwerp konstruktief hoofdsysteem. Voor de duidelijkheid is hier gekozen voor een eigen nummering, die te zien is in figuur 16.

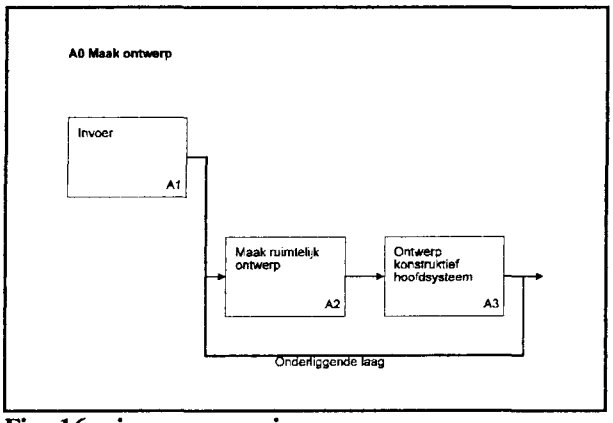


Fig. 16, eigen nummering

A2 Maak ruimtelijk ontwerp

Te zien in figuur 17.

Proces A21:

De plaats van een ruimte behorende bij een laag wordt vastgelegd.

Proces A22:

Plaats van volgende ruimte behorende bij een laag wordt vastgelegd zodat voorwaarden gelden.

Proces A23:

Is een ruimte eenmaal op een laag geplaatst dan wordt gekeken of de konstruktieonderdelen op de onderliggende laag wel voldoende aanleiding geven voor die bewuste plaatsing. De uitkomst van deze controle stuurt het proces A22, waardoor eventueel voor een nieuwe plaatsing gekozen kan worden. Het eindresultaat is een ruimtelijk ontwerp voor een laag.

Input I1:

Input I1 is de lijst van eventuele al aanwezig konstruktie-elementen. De plaatsing van een ruimtelijk element kan namelijk afhankelijk zijn van de al aanwezige konstruktie-elementen.

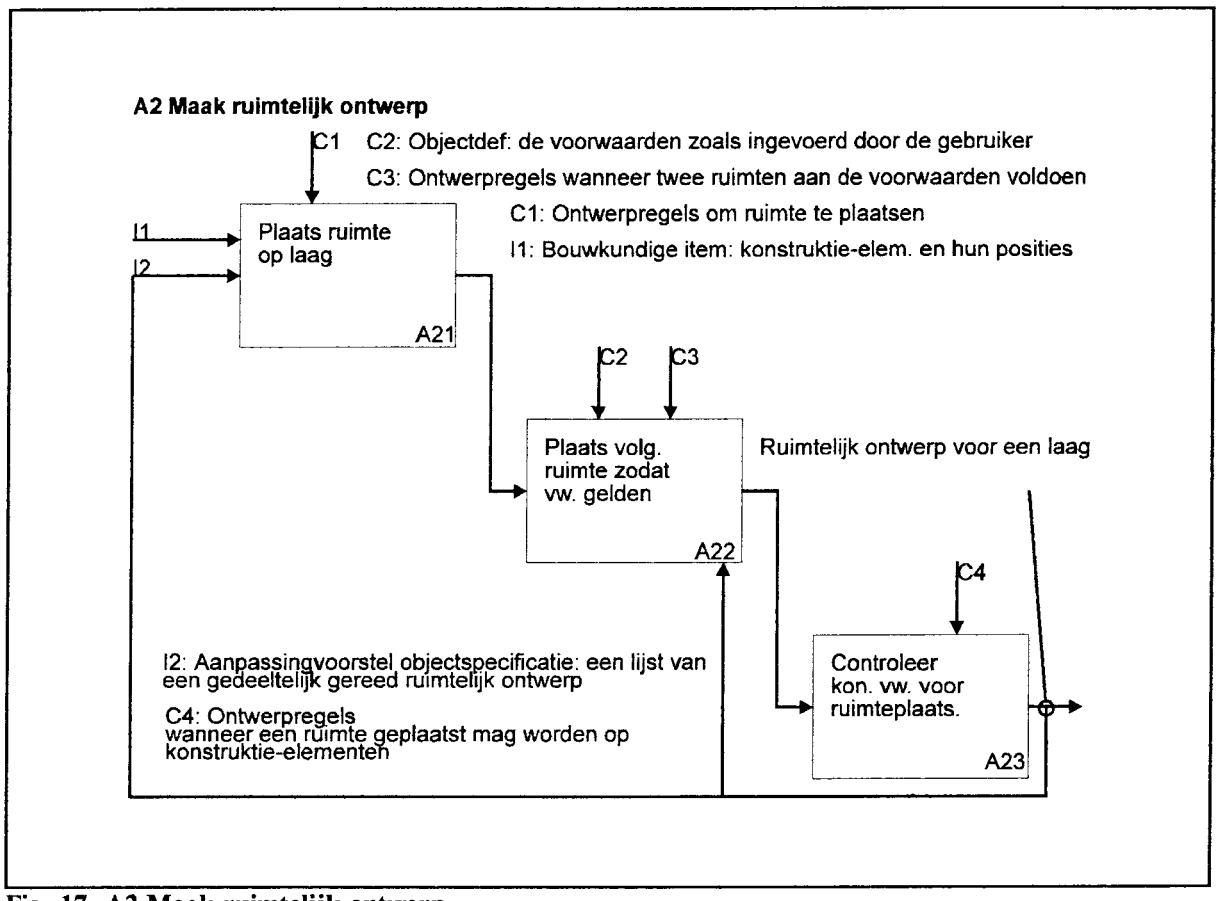


Fig. 17, A2 Maak ruimtelijk ontwerp

Input I2:

Om een ruimte te plaatsen is er soms reeds sprake van een ten dele gereed ruimtelijk ontwerp. Hiervoor is input I2, de lijst waarin het ten dele gerepresenteerde ruimtelijk ontwerp vermeld staat. Deze lijst kan natuurlijk ook leeg zijn, dan is er sprake van een nog niet aanwezig gedeeltelijk ruimtelijk ontwerp.

Ontwerpregels C1, C2 en C3:

Eerst, op een laag zoekniveau, vertellen deze regels hoe een ruimte het best geplaatst kan worden als er reeds andere ruimten geplaatst zijn en er voorwaarden bekend zijn die een verband hebben tussen de te plaatsen ruimte en de reeds geplaatste ruimten. Later, op een hoog zoekniveau, vertellen deze regels alleen, declaratief, hoe voorwaarden vervuld worden, maar niet, let wel, hoe deze voorwaarden bereikt worden! . Er zijn hiervoor controleregels die beslissen wanneer een ruimte op de onderliggende konstruktiedelen geplaatst mag worden en wanneer niet.

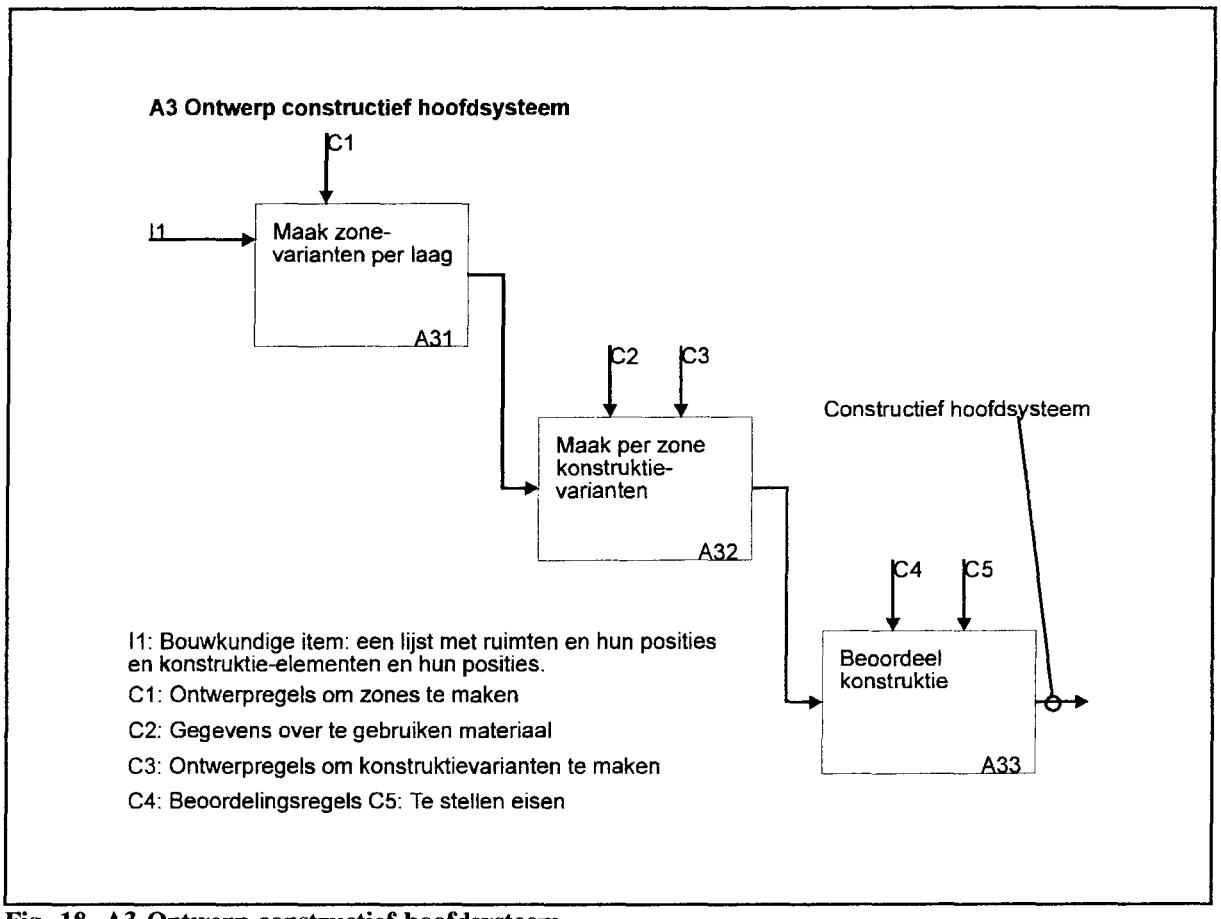


Fig. 18, A3 Ontwerp constructief hoofdsysteem

Objectdefinitie C2:

Deze objectdefinitie bevat de voorwaarden die vertellen welke ruimten met elkaar verbonden zijn, of ruimten een bepaalde onderlinge afstand nodig hebben, of ruimten boven elkaar mogen liggen of juist niet, enz.

De processen A21, A22 en A23 kunnen niet verder uitgewerkt worden.

A3 Ontwerp constructief hoofdsysteem

Te zien in figuur 18.

Proces A31:

Vanuit de ruimtelijke structuur wordt een zonering gemaakt van de ruimten: er worden ruimten geclusterd zodat grotere gehelen onstaan die gekonstrueerd worden. Het proces A31 Maak zonevarianten per laag wordt hierna nog verder uitgewerkt met IDEF0.

Proces A32:

Per zone worden een aantal konstruktievarianten uitgewerkt. Dit gebeurt niet zomaar: dit zal te zien zijn bij de uitwerking van het proces A32 Maak per zone konstruktievarianten. Op het moment dat voor een konstruktievariant gekozen wordt, moet deze variant nog ingepast worden in de zone. Dat wil zeggen dat de konstruktie-elementen een plaats moeten krijgen (onafhankelijk van de representatie van de zone) en een dimensionering.

Proces A33:

Vervolgens moet de konstruktie beoordeeld worden. De output van dit proces is een konstruktief hoofdsysteem per laag.

Input I1:

Een lijst met ruimten en hun posities en konstruktie-elementen en hun posities.

Ontwerpregels C1:

De gebruiker kan zijn voorkeur uitspreken voor een zone waarin geen lege plekken zitten of een geheel gevulde zone, enz.

Objectdefinitie C2:

Bij de objectdefintie C2 kunnen we de gebruiker laten kiezen voor een bepaald materiaalgebruik of het negeren van bepaalde konstruktievarianten, bijvoorbeeld omdat ze te duur zijn, of esthetisch niet goed worden bevonden.

Ontwerpregels C3:

Regels om konstruktievarianten te maken.

Ontwerpregels C4:

Regels om het programma van eisen te koppelen aan de gerealiseerde konstruktie.

Ontwerpregels C5:

Eisen uit Programma van Eisen.

A31 Maak zonevarianten per laag

Te zien in figuur 19.

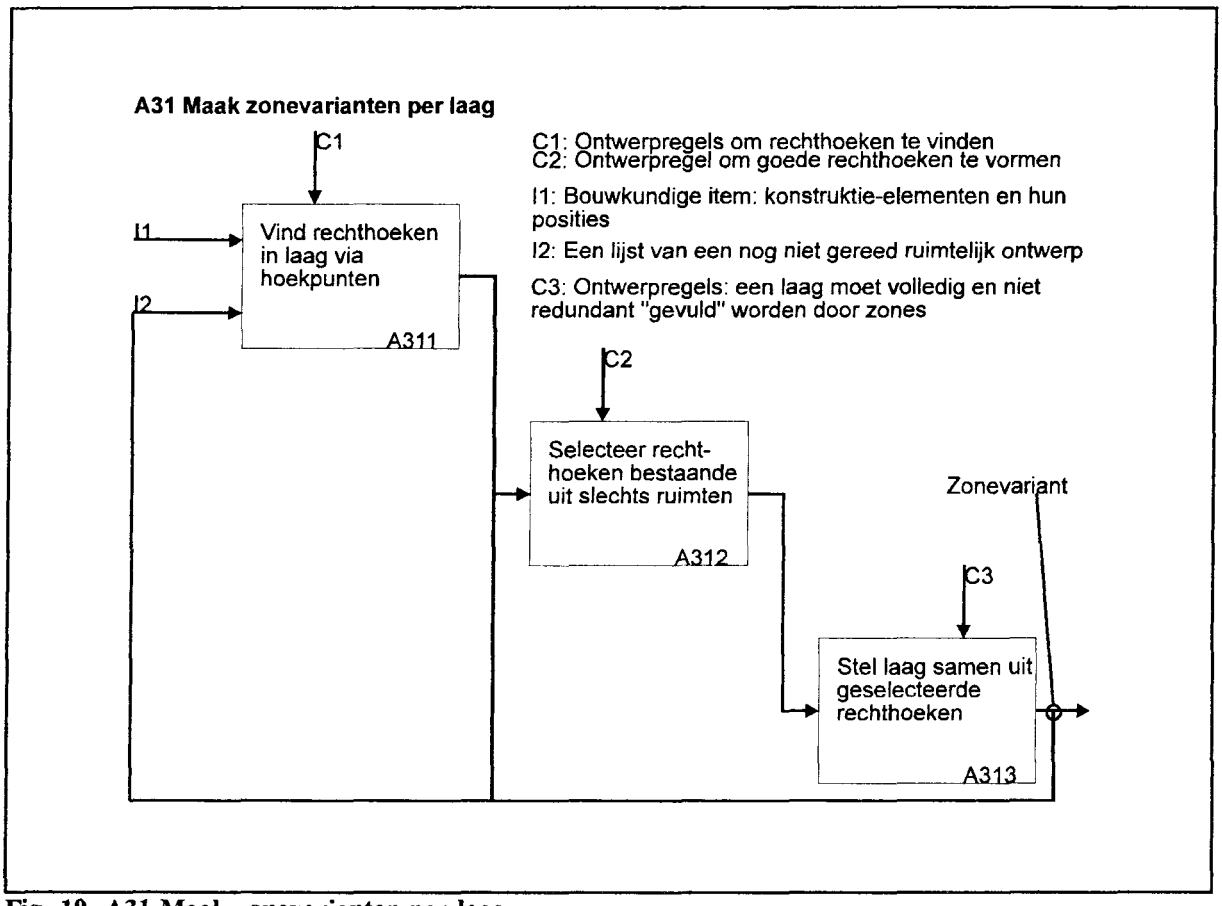


Fig. 19, A31 Maak zonevarianten per laag

Proces A311:

In de ruimtelijke laag, via een konstruktie van de hoekpunten van de afzonderlijke ruimten, worden rechthoeken gezocht.

Proces A312:

De rechthoeken die bestaan uit slechts ruimten worden geselecteerd. Dit is geen noodzakelijke procesgang maar een keuze. Extra onderzoek zou nodig zijn om te bepalen hoe een algoritme zou uitpakken met andere randvoorwaarden. Het is wel mogelijk om de ontwerpregels zelf te bepalen. Deze ontwerpregels zouden dan ook een lege verzameling kunnen bevatten of een aantal regels die precies vertellen wanneer een rechthoek goed wordt geacht en wanneer niet.

Proces A313:

Er wordt een plattegrond samengesteld uit de goedgekeurde rechthoeken. Het eindproduct is een zonevariant die wordt opgeslagen in het aanpassingsvoorstel objectspecificatie. Vervolgens kan het proces via A311 en A312 een aantal keren herhaald worden om alle varianten te genereren.

Ontwerpregel C1:

Legt uit hoe rechthoeken kunnen worden gevonden.

Ontwerpregel C2:

Regel om rechthoeken te vormen die slechts bestaan uit ruimten en waarvan de rechthoekzijden geheel bestaan uit randen van ruimten.

Ontwerpregel C3:

Regel die uitlegt hoe een laag geheel opgebouwd moet worden met zones, zonder dat twee of meer zones eenzelfde ruimte bevatten.

Input I1:

Lijst met konstruktie-elementen en hun posities.

Input I2:

Lijst met een gedeeltelijk gereed ontwerp.

A32 Maak per zone konstruktievarianten

Te zien in figuur 20.

Proces A321:

Er worden per zone in de laag de verhoudingen van de lengte en breedte bepaald en de absolute afmetingen. Vooral de correlatie tussen deze beide gegevens zijn van wezenlijk belang.

Proces A322:

Er wordt een passend konstruktiesysteem gezocht bij de bij A321 gevonden voorwaarden. Het omzetten van het gekozen konstruktiesysteem naar een aantal elementen met afmetin-

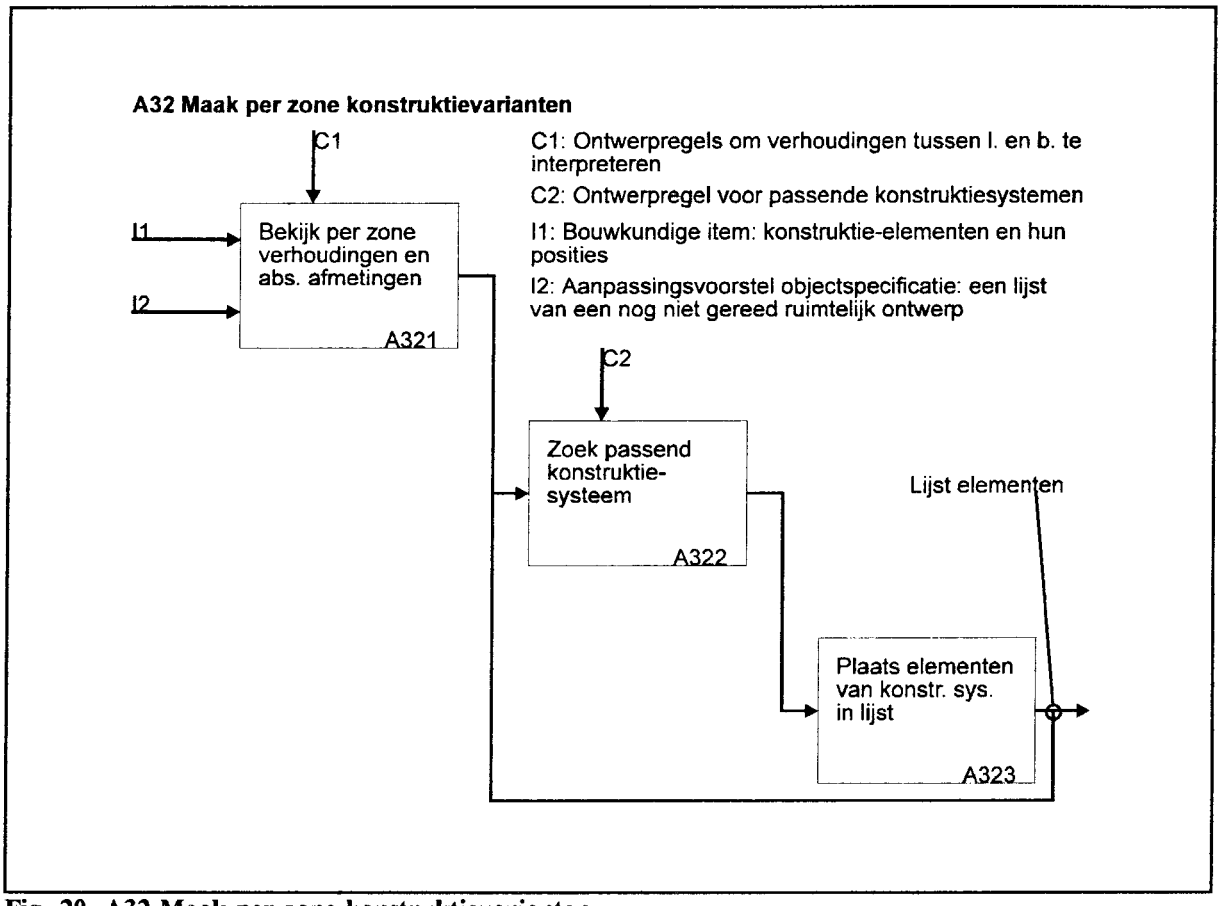


Fig. 20, A32 Maak per zone konstruktievarianten

gen vindt plaats in proces A322. Omdat per konstruktiesysteem dit proces zeer afwijkend plaatsvindt, is het moeilijk, zoniet onmogelijk hier een processchema van te maken.

Proces A323:

De elementen van dit konstruktiesysteem worden in de lijst "konstruktie-elementen en hun posities" en vervolgens worden eventuele variant-konstruktiesystemen bekeken. Het proces door A322 en A323 wordt enige malen doorlopen.

Ontwerpregels C1:

Regels om de verhoudingen tussen lengte en breedte van ruimten te interpreteren en hierdoor een inzicht te krijgen in het type ruimte.

Ontwerpregels C2:

Regels die aangeven welke konstruktiesystemen geschikt zijn voor welke type ruimten en welke konstruktie-elementen deze konstruktiesystemen bevatten.

Input I1:

Lijst met konstruktie-elementen en hun posities.

Input I2:

Lijst met gedeeltelijk gereed ruimtelijk ontwerp.

4.5.2 Datamodel Konstruktief Ontwerpen

Van het algoritme kan een gegevensmodel gemaakt worden in IDEF1x. Een gegevensmodel bevat entiteiten: "iets" dat een betekenis heeft voor de gebruiker en waarover het wenselijk is gegevens vast te leggen. Entiteiten worden, zoals reeds bij paragraaf 4.2 verteld, aangegeven door rechthoeken. Boven deze rechthoeken wordt de naam van de entiteit geschreven. In de rechthoeken worden attributen van de entiteit weergegeven. Een attribuut is een ondeelbaar beschrijvend kenmerk van een entiteit, waarvan de specifieke waarde kan worden verbonden met individuele entiteiten. Relaties tussen entiteiten worden aangegeven met een lijn met op het uiteinde een zwarte cirkel waarbij de cardinaliteit wordt aangegeven. Een relatie is een gezamenlijke eigenschap van entiteiten die ze afzonderlijk niet hebben. De cardinaliteit geeft aan hoeveel entiteiten in relatie staan met de andere entiteit. Eerst wordt gekeken naar het datamodel voor het algoritmedeel ruimtelijk ontwerp. Zie figuur 21. De hierbij behorende omschrijving, volgens het BIM, is hieronder gegeven.

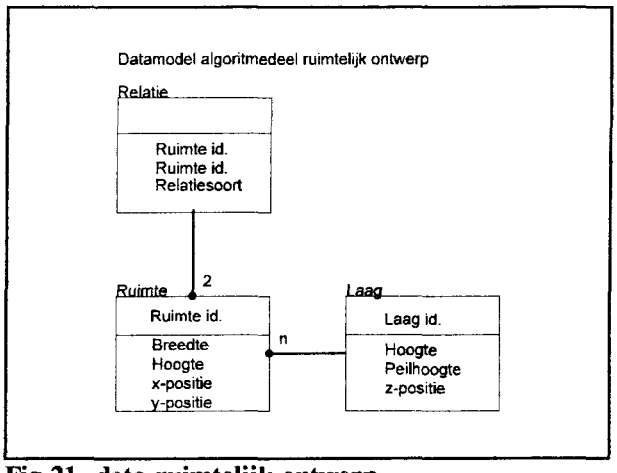


Fig 21, data ruimtelijk ontwerp

Entiteit: **Ruimte**

een rechthoekige geconditioneerde afbakening van ruimte waarin activiteiten kunnen plaatsvinden

Attributen

Ruimte.id
Breedte
Hoogte
x-positie
y-positie

Attribuutomschrijving

Ruimte.id

de naam van een ruimte gegeven door de gebruiker

Breedte

indien de ruimte op een ondergrond wordt gezet met een x-y-assenstelsel is de breedte het verschil tussen de uiterste x-waarden van hoekpunten van deze ruimte

Hoogte

indien de ruimte op een ondergrond wordt gezet met een x-y-assenstelsel is de hoogte het verschil tussen de uiterste y-waarden van hoekpunten van deze ruimte

x-positie

de x-positie van het middelpunt op de "vloer" van de ruimte op het x-y-assenstelsel waarop de ruimte geplaatst wordt

y-positie

de y-positie van het middelpunt op de "vloer" van de ruimte op het x-y-assenstelsel waarop de ruimte geplaatst wordt

Entiteit: **Relatie**

geeft aan of er een relatie tussen 2 ruimten bestaat, indien de relatie tussen 2 ruimten niet bestaat, is er geen relatie tussen deze 2 ruimten

Attributen

Ruimte.id
Ruimte.id

Relatiesoort

Attribuutomschrijving

Ruimte.id

de naam van een ruimte gegeven door de gebruiker

Relatiesoort

de soort relatie, de relatie tot nu toe gebruikt is "aanliggend", die aangeeft dat twee ruimten bij elkaar moeten liggen

Entiteit: **Laag**

een verdieping van een gebouw waarin diverse ruimten kunnen liggen

Attributen

Laag.id
Hoogte
Peilhoogte (z-positie)

Attribuutomschrijving

Laag.id

de naam van een laag gegeven door het verdiepingsnummer, hierdoor is de volgorde van de lagen bekend

Hoogte

de hoogte van een laag, vergelijkbaar met de verdiepingshoogte

Peilhoogte

de hoogte waarop een laag ligt ten opzichte van de begane grond of het bouwpeil

Het volgende te behandelen datamodel handelt over het algoritmedeel dat de ruimtelijke ontwerpen zoneert. Het datamodel is te zien in figuur 22.

De entiteiten Ruimte en Laag zijn reeds behandeld.

Entiteit: **Zone**

een rechthoekige begrenzing van ruimte op 1 verdieping/laag bestaande uit 1 of meerdere ruimten

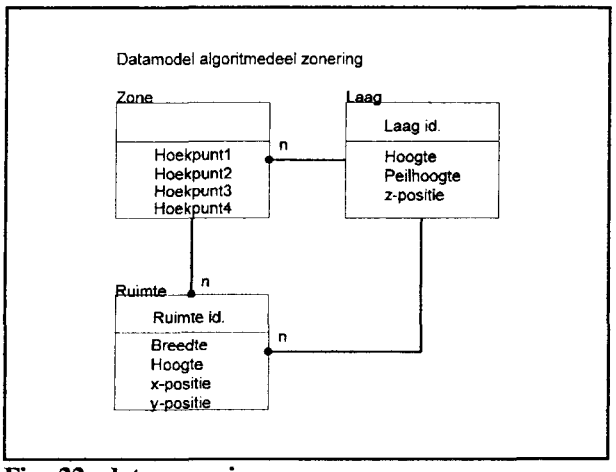


Fig. 22, data zonering

Attributen

Hoekpunt1
Hoekpunt2
Hoekpunt3
Hoekpunt4

Attribuutomschrijving

Hoekpunt1

linkerbovenhoekpunt van plattegrond zone gezien vanaf bovenkant zone

Hoekpunt2

rechterbovenhoekpunt van plattegrond zone gezien vanaf bovenkant zone

Hoekpunt3

rechteronderhoekpunt van plattegrond zone gezien vanaf bovenkant zone

Hoekpunt4

linkeronderhoekpunt van plattegrond zone gezien vanaf bovenkant zone

Vervolgens kan het datamodel bekeken worden van het algoritmedeel konstruktie, te zien in figuur 23.

Entiteit: **Draagstruktuur**

wordt gekozen afhankelijk van de absolute afmetingen en verhoudingen van de afmetingen van een zone en bestaat uit informatie hoe een zone gekonstrueerd wordt, bijvoorbeeld "kolommen op de hoeken met een ruimtevakwerk" of "schijven langs de wanden en betonplaten hierop"

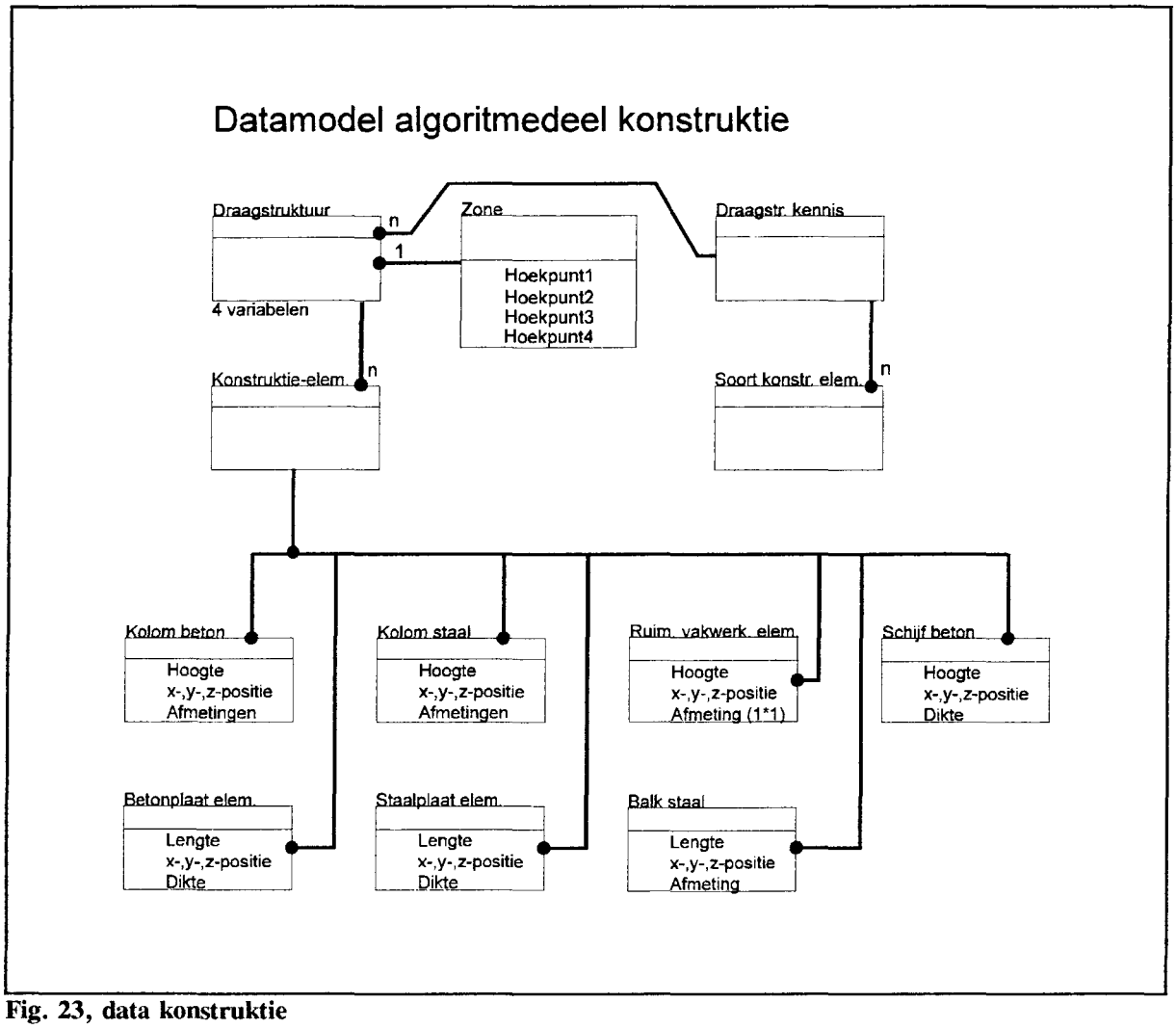


Fig. 23, data konstruktie

Entiteit: **Konstruktie-elem.**

een verzameling van konstruktie-elementen

Entiteit: **Draagstr. kennis**

de kennis welke draagstrukturen toepasbaar zijn bij zones met bepaalde afmetingen en verhoudingen van afmetingen

Entiteit: **Soort konstr. elem.**

de kennis welke soort konstruktie-elementen te gebruiken zijn bij een type draagstruktuur

Entiteit: **Kolom beton**

een betonnen vierkante kolom te gebruiken om een zone te konstrueren, wordt telkens binnen de ruimtelijke grenzen van de ruimte geplaatst

Attributen

x-,y-,z-positie
Hoogte
Afmetingen

Attribuutomschrijving

x-,y-,z-positie

de x-,y- en z-positie van een punt van de kolom, zie voor de plaats van dit punt de programmabeschrijving in de bijlagen, de kolom wordt geheel binnen de ruimtelijke grenzen geplaatst

Hoogte

hoogte van de kolom, de kolom wordt niet hoger dan de hoogte van de laag waarin de kolom geplaatst wordt, omdat de kolom geheel binnen de ruimtelijke grenzen geplaatst wordt

Afmetingen

de breedte van de (vierkante) kolom, wordt aangenomen als 1/20 van de hoogte van de kolom

Entiteit: **Kolom staal**

een stalen geprofileerde kolom te gebruiken om een zone te konstrueren, wordt telkens binnen de ruimtelijke grenzen van de ruimte geplaatst

Attributen

x-,y-,z-positie
Hoogte
Afmetingen

Attribuutomschrijving

x-,y-,z-positie

de x-,y- en z-positie van een punt van de kolom, zie voor de plaats van dit punt de programmabeschrijving in de bijlagen, de kolom wordt geheel binnen de ruimtelijke grenzen geplaatst

Hoogte

hoogte van de kolom, de kolom wordt niet hoger dan de hoogte van de laag waarin de kolom geplaatst wordt, omdat de kolom geheel binnen de ruimtelijke grenzen geplaatst wordt

Afmetingen

de hoogte van de doorsnede van de geprofileerde kolom, de breedte van de doorsnede is hieruit af te leiden omdat het gebruik van standaard profielen aangenomen wordt, wordt 1/20 van de hoogte genomen

Entiteit: **Ruim. vakwerk. elem.**

een deel van een ruimtelijk vakwerk bestaande uit een aantal staven tezamen een rechthoek vormend van 1 bij 1 meter

Attributen

Hoogte
x-,y-,z-positie
Afmetingen

Attribuutomschrijving

Hoogte

de konstruktiehoogte van een ruimtelijk vakwerk-element bepaald door de totale overspanning van het ruimtelijk vakwerk, volgens afspraak valt het ruimtelijk vakwerk geheel binnen de ruimtelijke vorm

x-,y-,z-positie

de x-,y- en z-positie van een punt van het element. Zie voor dit specifieke punt de bijlage met programmabeschrijving, volgens afspraak valt het ruimtelijk vakwerk geheel binnen de ruimtelijke vorm

Afmetingen

de voorlopig vastgestelde afmetingen van een element bedragen 1 bij 1 meter

Entiteit: **Schijf beton**

een schijfvormig konstruktie-element, bestaande uit beton, om een zone te konstrueren

Attributen:

Hoogte
Dikte
x-,y-,z-positie

Attribuutomschrijving

Hoogte

hoogte van de schijf, bepaald door de laag waarin de schijf zich bevindt, volgens afspraak valt de schijf geheel binnen de ruimtelijke vorm

Dikte

dikte van de schijf, bepaald door de hoogte van de laag waarin de schijf zich bevindt, volgens afspraak valt de schijf geheel binnen de ruimtelijke vorm

x-,y-,z-positie

x-,y- en z-positie van een punt van de schijf, zie voor dit punt de bijlage met programmabeschrijving, volgens afspraak valt de schijf geheel binnen de ruimtelijke vorm

Entiteit: **Betonplaat elem.**

een konstruktie-element om een zone te konstrueren bestaande uit een betonplaat

Attributen

Lengte
x-,y-,z-positie
Dikte

Attribuutomschrijving

Lengte

de lengte van een betonplaat afhankelijk van de zone waarvoor de betonplaat gebruikt wordt, volgens afspraak wordt de betonplaat op de ruimtelijke vorm gelegd en ligt dus eigenlijk, indien een ruimte boven de beschouwde zone ligt, in de ruimte boven de zone

x-,y-,z-positie

de x-,y- en z-positie van een punt van het element, voor dit punt wordt verwezen naar de bijlage met programma-omschrijving

Dikte

de dikte van de betonplaat afhankelijk van de overspanning waarvoor de betonplaat gebruikt wordt, volgens afspraak wordt de betonplaat op de ruimtelijke vorm gelegd

Entiteit: **Staalplaat elem.**

een konstruktie-element om een zone te konstrueren bestaande uit een geprofileerde staalplaat

Attributen

Lengte
x-,y-,z-positie
Dikte

Attribuutomschrijving

Lengte

de lengte van een staalplaat afhankelijk van de zone waarvoor de staalplaat gebruikt wordt, volgens afspraak wordt de staalplaat op de ruimtelijke vorm gelegd en ligt

dus eigenlijk, indien een ruimte boven de beschouwde zone ligt, in de ruimte boven de zone

x-,y-,z-positie

de x-,y- en z-positie van een punt van het element, voor dit punt wordt verwezen naar de bijlage met programma-omschrijving

Dikte

de dikte van de staalplaat afhankelijk van de overspanning waarvoor de staalplaat gebruikt wordt, volgens afspraak wordt de staalplaat op de ruimtelijke vorm gelegd

Entiteit: **Balk staal**

een stalen balk gebruikt om een zone te konstrueren in combinatie met veelal een aantal stalen kolommen, volgens afspraak ligt de balk geheel binnen de ruimtelijke vorm

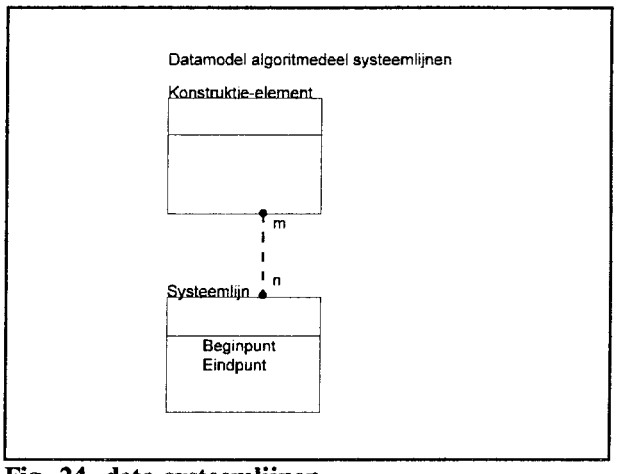


Fig. 24, data systeemlijnen

Attributen:

Lengte
x-,y-,z-positie
Afmeting

Attribuutomschrijving

Lengte

de lengte van de balk, afhankelijk van de afmetingen van de te konstrueren zone, volgens afspraak ligt de balk geheel binnen de ruimtelijke vorm

x-,y-,z-positie

de x-,y- en z-positie van een punt van de balk. Zie voor dit punt de bijlage met programma-omschrijving

Afmeting

de hoogte van de doorsnede van de balk, afhankelijk van de overspanning van de balk

Vervolgens is er nog een klein datamodel voor het algoritmedeel systeemlijnen, zie figuur 24. Uit de diverse gegevens van de konstruktie-elementen zijn systeemlijnen te berekenen, vandaar de stippellijn tussen de twee entiteiten die te zien zijn in de afbeelding.

Entiteit: **Systeemlijn**

een lijn die het centrum in een konstruktie-element aangeeft en mechanica-berekeningen mogelijk maakt

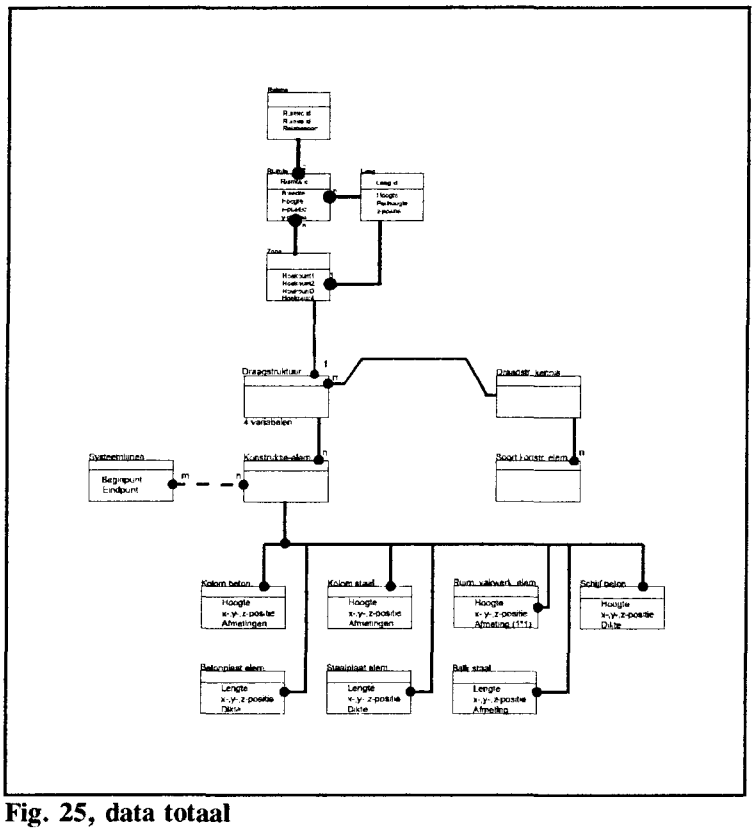


Fig. 25, data totaal

Attributen

Beginpunt
Eindpunt

Attribuutomschrijving

Beginpunt

beginpunt van systeemlijn met x-,y- en z-positie

Eindpunt

eindpunt van systeemlijn met x-,y- en z-positie

In figuur 20, is het gehele datamodel nog eens te zien. In de bijlage met afbeeldingen is dit figuur groter afgebeeld.

5 Processen en data omzetten in algoritmen

5.1 Inleiding

Nu tot een zekere hoogte duidelijk is hoe een konstruktief-ontwerp-proces verloopt kan geprobeerd worden algoritmen te maken om het konstruktief-ontwerp-proces te simuleren. Is het konstruktief-ontwerp-proces eenmaal gesimuleerd, dan is het ook mogelijk om dit proces bij te sturen en te veranderen. De gebruiker van het algoritme kan interactief met het proces werken. Hierdoor ontstaat een hulpmiddel voor het konstruktief-ontwerp-proces. Het vinden van algoritmen is in dit geval een bijna empirisch proces. Een algoritme wordt gemaakt op gevoel, gecorrigeerd zodat het werkelijk functioneert en getoetst. Hier volgt een beschrijving van de gevonden algoritmen. Een deel van het algoritme of algoritmen (het geheel kan als een enkele algoritme gezien worden maar het geheel kan ook gezien worden als opgebouwd uit diverse algoritmen) bestaat uit het genereren van ruimtelijke vormen, een simulatie van het ruimtelijk-ontwerp-proces. Hoewel dit deelalgoritme in dit afstudeerproject gemaakt is zijn vele overeenkomsten zichtbaar met en veel details overgenomen van bestaande space-allocation-algoritmen.

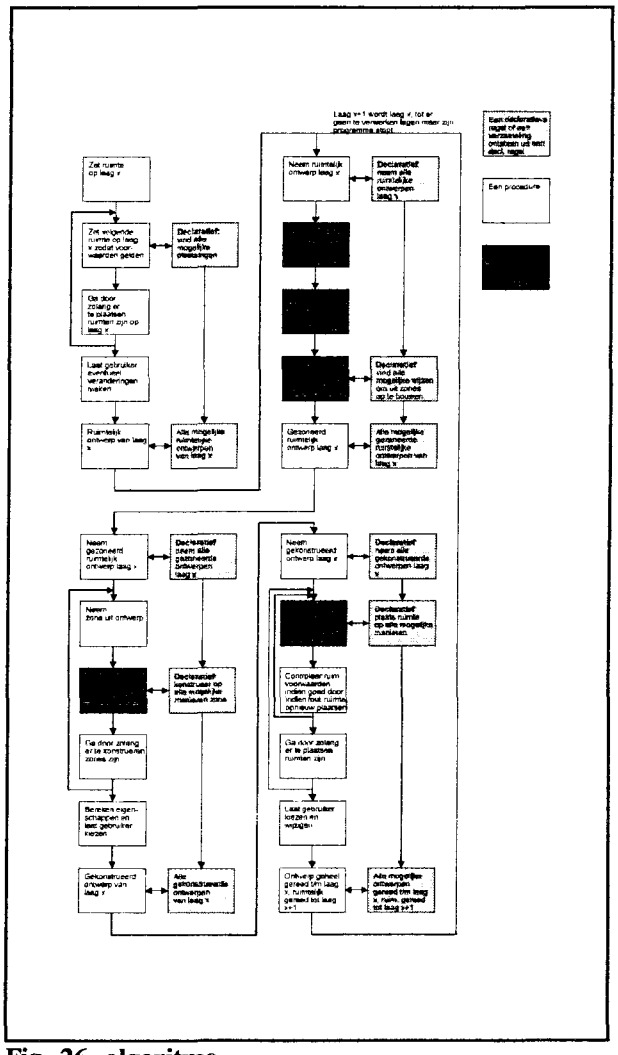


Fig. 26, algoritme

5.2 Formele beschrijving algoritme

Eerst zal een flowchart besproken worden van het programma, al zal duidelijk worden dat deze niet geheel toepasbaar is omdat gebruik wordt gemaakt van een declaratieve taal. Deze flowchart is als illustratie te zien in figuur 26 en verder vergroot weergegeven in de figurenbijlage.

Formele beschrijving van de processen

Opmerking: overal waar "op een laag" staat wordt niet bedoeld dat iets fysiek op de laag aanwezig is, maar dat iets onderdeel is van een laag of deel uit maakt van een laag.

Deelalgoritme ruimtelijk ontwerp

Proces: Invoer

Wordt niet besproken, zie voor details de bijlage "Gebruiksaanwijzing, code en toelichting bij programma" en 5.3.

Proces: Zet ruimte op laag x

Een rechthoekige ruimte met lengte en breedte wordt op (hiermee wordt eigenlijk "in" bedoeld) laag x (hier is dat altijd de eerste laag) geplaatst met bepaalde hoogte op een stramien van 1 bij 1 meter. De moduulmaat is eveneens 1 meter. Het middelpunt van de ruimte is het plaatsingspunt, zie figuur 28.

Proces: Zet volgende ruimte op laag x zodat voorwaarden gelden

Rechthoekige ruimte met lengte en breedte wordt op een laag geplaatst met zodat de voorwaarden gelden die bij de invoer ingegeven zijn. Dat wil zeggen dat ruimten aanliggend geplaatst moeten worden. Bij de moduulmaat en rastermaat van 1 meter is het duidelijk dat de minimale aanliggende lengte tussen twee ruimten 1 meter is, zie figuur 27, 29.

Proces: Ga door zolang er te plaatsen ruimten zijn op laag x

Alle ruimten die volgens de invoer op een laag geplaatst moeten worden dienen geplaatst te worden. Zodoende wordt het plaatsingsproces enkele malen herhaald, zie figuur 30.

Proces: Laat de gebruiker eventueel veranderingen maken

Wordt niet besproken, zie voor details de bijlage "Gebruiksaanwijzing, code en toelichting bij programma".

Deze processen vormen tezamen het ruimtelijk ontwerpproces, zie voor een minder formele beschrijving 5.4. Declaratief wordt verteld dat bij het proces "Zet volgende ruimte op laag x zodat voorwaarden gelden" alle mogelijke plaatsingen moeten worden gezocht. Volgens een verder niet bekende procedure worden alle mogelijke plaatsingen gezocht met als resultaat alle mogelijke ruimtelijke ontwerpen van laag x.

Deelalgoritme zonering

Proces: Neem ruimtelijk ontwerp x

Een ruimtelijk ontwerp uit de verzameling van alle mogelijke ruimtelijke ontwerpen van laag x wordt genomen.

Proces: Vind uit verzameling hoekpunten ruimten alle rechthoeken

Op de laag liggen ruimten. De hoekpunten van de plattegronden van de ruimten wordt als verzameling punten beschouwd. Uit deze verzameling zijn nieuwe rechthoeken te vormen. Een declaratief proces (dus niet goed in een flowchart te tekenen) zorgt ervoor dat alle rechthoeken gevonden worden, zie figuur 35.

Proces: Selecteer rechthoeken die zones zijn

Bepaalde gevonden rechthoeken sluiten enkele plattegronden van ruimten op de laag precies in, waarbij de rechthoek niet geheel gevuld behoeft te zijn met ruimten. Een declaratief proces selecteert rechthoeken die zones zijn, zie figuur 33.

Proces: Bouw ruimtelijk ontwerp laag x op uit zones

Laag bestaande uit ruimten wordt "volgelegd" met zones, te zien in figuur 38. Elke ruimte moet bevat zijn in zone en een ruimte mag niet bevat worden door 2 of meer zones. Dit wordt gerealiseerd door een declaratieve beschrijving.

Deze processen vormen tezamen het zoneringsproces. Zie voor een minder formele beschrijving 5.5. Een declaratieve regel zorgt ervoor dat alle ruimtelijke ontwerpen uiteindelijk gezoneerd worden en verder dat alle mogelijke wijzen om uit zones ruimtelijke ontwerpen van de laag op te bouwen worden verwerkt tot een verzameling van alle mogelijke gezoneerde ruimtelijke ontwerpen van de laag x.

Deelalgoritme konstruktie

Proces: Neem gezoneerd ruimtelijk ontwerp laag x

Een gezoneerd ruimtelijk ontwerp uit de verzameling van gezoneerde ruimtelijke ontwerpen van laag x.

Proces: Neem zone uit ontwerp

Een zone van de laag x wordt genomen.

Proces: Konstrueer zone: maak elementen en stramienlijnen aan

Bij bepaalde afmetingen zones horen bepaalde konstruktiesystemen. Aan de hand van declaratieve regels worden konstruktie-elementen en stramienlijnen gegenereerd. Een

declaratieve regel zorgt er verder voor dat alle mogelijke konstruktiemethoden ook aan bod komen bij een volgende variant, zie voor een voorbeeld figuur 39.

Proces: Ga door zolang er te konstrueren zones zijn

Alle zones van de laag x dienen gekonstrueerd te worden. Vanaf het proces "Neem zone uit ontwerp" wordt het algoritme een paar maal herhaald.

Proces: Bereken eigenschappen en laat gebruiker kiezen

Het gewicht van de delen staal en beton van de konstruktie-elementen wordt berekend van de tot dan toe gekonstrueerde hoeveelheid lagen (dus tot en met laag x). De keuzes van de gebruiker zijn te vinden in de bijlage "Gebruiksaanwijzing, code en toelichting bij programma".

Proces: Gekonstrueerd ontwerp van laag x

Nu is een gekonstrueerd ontwerp van laag x aanwezig. Declaratieve regels hebben ervoor gezorgd dat de hele verzameling mogelijke gekonstrueerde ontwerpen van laag x aanwezig is.

Deze processen tezamen vormen het konstruktieproces, minder formeel en uitgebreider beschreven in 5.6.

Deelalgoritme ruimtelijk ontwerp met voorwaarden

Proces: Neem gekonstrueerd ontwerp laag x

Er wordt een gekonstrueerd ontwerp van laag x genomen.

Proces: Plaats ruimte op laag x+1 afh. van konstr. elem. laag x

Afhankelijk van waar en hoeveel konstruktie-elementen op de laag x aanwezig zijn wordt ergens een ruimte die op laag x+1 geplaatst moet worden geplaatst.

Proces: Controleer ruim. voorwaarden, indien goed door, indien fout ruimte opnieuw plaatsen

Als de ruimte op laag x+1 een plaats heeft gekregen moet worden gekeken of deze ruimte andere ruimten op deze laag niet overlapt en of aan de ruimtelijke voorwaarden wat betreft aanligging van andere ruimten wordt voldaan. Is dit niet het geval, dan dient de ruimte op een andere plaats te worden geplaatst. Is dit wel het geval, dan kan door worden gegaan met het volgende proces. De plaatsingsregels zijn declaratief.

Proces: Ga door zolang er te plaatsen ruimten zijn

Omdat alle ruimten die geplaatst moeten worden op laag x+1 behandeld dienen te worden wordt vanaf proces "Plaats ruimte op laag x+1 afh. van konstr. elem. laag x" de procedure enkele malen herhaald.

Proces: Laat gebruiker kiezen en wijzigen

Zie de bijlage "Gebruiksaanwijzing, code en toelichting bij programma".

Deze processen tezamen vormen het deelalgoritme voor het ruimtelijke ontwerp met voorwaarden, verder uitgewerkt in 5.8.

Vervolgens worden de deelalgoritmen voor zonering, konstruktie en ruimtelijk ontwerp met voorwaarden enkele malen doorlopen om alle lagen te verwerken.

Bij nauwkeurige beschouwing van de flowchart in figuur 26, is te zien dat naast procedures de declaratieve regels een belangrijke rol op zich nemen. Behalve het feit dat enkele schijnbare procedures declaratief uitgewerkt zijn (er is dus alleen verteld wat de procedures moeten bereiken, en niet wat ze moeten doen om dat te bereiken) is de declaratieve programmeerwijze ook gebruikt om gemakkelijk alle varianten die aan bepaalde voorwaarden voldoen te genereren. Overigens is het geheel declaratief geprogrammeerd omdat ook procedureel programmeren met een declaratieve taal mogelijk is. De voordelen van deze hybride programmeerwijze zijn de volgende:

- Overzichtelijke en gebruikelijke procedurele hoofdstructuur
- Moeilijk procedureel te programmeren gedeelten zijn declaratief te programmeren
- Het gedeeltelijke declaratieve gedeelte zorgt voor het gemakkelijk genereren van varianten

Er wordt nu begonnen met een minder formele en uitgebreide beschrijving van de algoritmen.

5.3 Invoer

De invoer voor het algoritme bestaat tot nu toe uit de volgende zaken:

De naam, de breedte en de lengte van een ruimte, de laag waarop de ruimte ligt

Zo is bijvoorbeeld een invoer: ruimte(keuken,2,4,1). Dit is weliswaar een concrete invoer bij het programma, maar er is duidelijk te zien wat de invoer is. De ruimte heeft een naam "keuken", een breedte van 2 meter en een lengte van 4 meter. Verder ligt de ruimte

"keuken" op de eerste laag. In dit geval moet het woord laag niet te serieus worden genomen: men kan hiervoor in gedachte de begane grond van een gebouw nemen.

Voorwaarden die een relatie leggen tussen verschillende ingevoerde ruimten, of die een ruimte bepaalde beperkingen oplegt

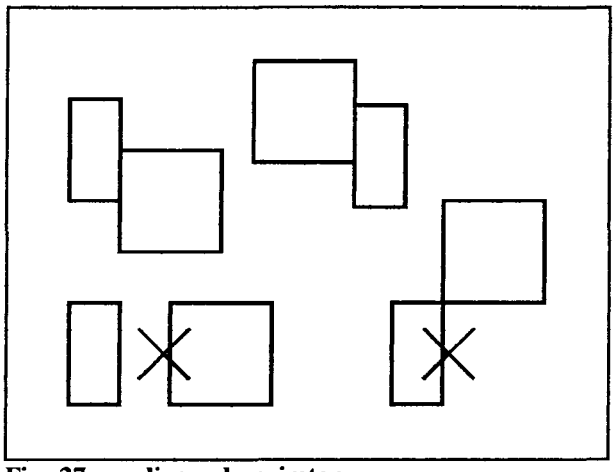


Fig. 27, aanliggende ruimten

Tot nu toe wordt slecht 1 relatie gebruikt, namelijk de relatie "aanliggend". De relatie "aanliggend" wordt gebruikt, in combinatie met twee ruimten, om aan te geven dat de twee ruimten aan elkaar liggen. In figuur 27 zijn enkele voorbeelden gegeven van ruimten die aan elkaar liggen. Een voorbeeld van de invoer van de relatie aanliggend zou kunnen zijn: vw(a,a,keuken). Er is een voorwaarde (door het predicaat "vw"). De eerste "a" in het predicaat staat voor "aanliggend", terwijl de tweede "a" in het predicaat staat voor ruimte "a". Het woord "keuken" staat voor de ruimte "keuken". De ruimte "keuken" is al bij het invoeren van de gegevens over een ruimte in de vorige alinea besproken. Voorwaarden die een relatie leggen tussen verschillende ingevoerde ruimten zijn facultatief. Het zou best zo kunnen zijn dat het niet nodig behoeft te zijn dat er ook maar 1 ruimte aansluit op een andere ruimte.

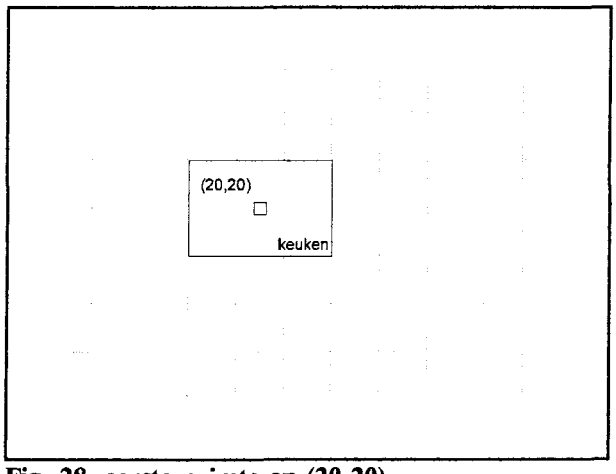


Fig. 28, eerste ruimte op (20,20)

Informatie over de lagen

Lagen worden in het algoritme "lagen" genoemd. Er werd reeds ingevoerd op welke laag een ruimte lag. De ruimte "keuken" lag op laag 1. De informatie over deze lagen hoort ook bij de invoer op dit moment. Er wordt ingevoerd hoe hoog de laag is en op welke peilhoogte de laag: de afstand tussen de begane grond en de vloer van de laag. Een voorbeeld hiervan zou kunnen zijn: laag(1,2,0). Dit is weliswaar een concrete invoer bij het algoritme, maar er is duidelijk te zien wat de invoer is. Het eerste cijfer "1" geeft aan dat gesproken wordt over de eerste laag. Het cijfer "2" geeft aan dat de laag 2 meter hoog is. Het cijfer "0" geeft de peilhoogte aan, deze blijkt in het voorbeeld "0" te zijn.

Kort samengevat is de invoer:

Invoer

Ruimte: **Naam**
Lengte
Breedte
Naam laag

Laag: **Naam**
Hoogte
Peilhoogte

Facultatieve invoer

Voorwaarden: **Naam voorwaarde (nu "a")**
Afhankelijk van voorwaarde: naam van een of meer ruimten (nu 2)

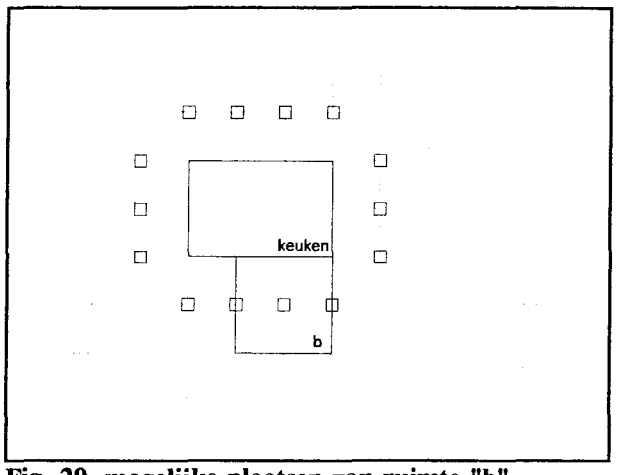


Fig. 29, mogelijke plaatsen van ruimte "b"

Elke ruimte dient minstens aan een andere ruimte gekoppeld te zijn op dezelfde laag, zodat geen ruimten geen enkele relatie met een andere ruimte hebben. Alle andere extra voorwaarden zijn facultatief.

5.4 Plaatsing van ruimten

Wat gaat het algoritme nu allemaal met deze invoer doen? Allereerst neemt het algoritme de ruimten apart die op de eerste laag (in het algoritme gebruikten we het woord "laag" voor verdieping) moeten liggen. Het algoritme gaat deze ruimten zo plaatsen dat aan de eventueel gestelde voorwaarden voldaan zal worden. Impliciet wordt er vanuit gegaan dat de ruimten rechthoekig zijn en eenduidig bepaald door hun lengte en breedte, d.w.z. dat ze niet geroteerd mogen worden. Het plaatsen van de ruimten op de eerste laag wordt als volgt gedaan. Er wordt een willekeurige ruimte als eerste beet genomen. Deze ruimte wordt geplaatst op een punt in het midden van een raster. Dit punt heeft de x- en y-waarden (20,20). Even goed had

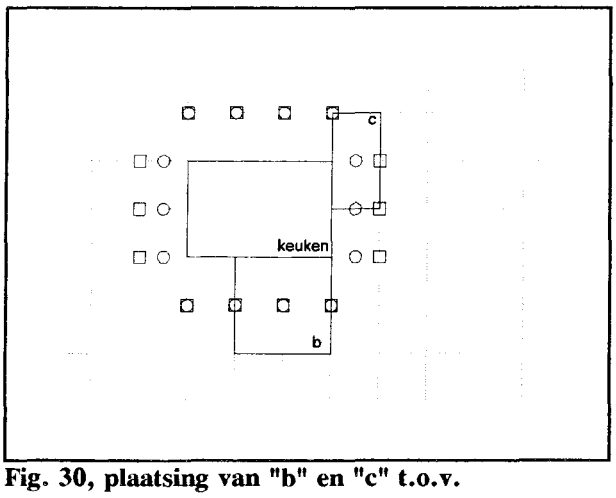


Fig. 30, plaatsing van "b" en "c" t.o.v. "keuken"

een andere waarde voor de x- en y-waarden genomen kunnen worden, maar er moet ergens begonnen worden. Dit is te zien in figuur 28. Vervolgens wordt gekeken of er een voorwaarde is waarin de geplaatste ruimte genoemd wordt. Is de geplaatste ruimte op (20,20) bijvoorbeeld ruimte "keuken" dan wordt gezocht naar een voorwaarde waarin de ruimte "keuken" genoemd wordt. Als er geen ruimte bestaat die samen met de ruimte "keuken" in een voorwaarde "aanliggend" staat, dan is in beginsel de ruimte "keuken" positieloos, want de ruimte heeft geen enkele relatie met andere ruimten. Het is dan ook duidelijk dat het oplossingenaantal oneindig zal zijn. Ruimte "keuken" is namelijk overal te plaatsen zonder dat tegen de voorwaarden gezondigd wordt. Overal een ruimte kunnen plaatsen in een oneindig raster betekent dat er oneindig veel oplossingen zijn. Stel nu dat de voorwaarde waarin de ruimte "keuken" bestaat en de andere ruimte die genoemd wordt in de voorwaarde de ruimte "b" is. Dan zal de ruimte "b" de volgende ruimte zijn die geplaatst wordt op het raster. De ruimte "b" zou aansluiten op de ruimte "keuken" op punt (20,20). Nu wordt een lijst gemaakt van alle punten waarop ruimte "b" kan liggen zodat ruimte "b" aan ruimte "keuken" ligt. Deze lijst van punten is beperkt. Er worden discrete plaatsen gemaakt om de 1 meter. Er onstaat dus een raster van punten waar ruimten geplaatst kunnen worden. Een voorbeeld hiervan is te zien in figuur 29. Er is nu wellicht een aantal punten waarop ruimte "b" geplaatsen kan worden zodat ruimte "b" aansluit bij ruimte "keuken". Laat ruimte "b" dan maar gewoon op een van deze punten geplaatst worden. Stel dat er nog meer ruimten liggen op de eerste laag, bijvoorbeeld ruimte "c", en dat verder nog ingevoerd is dat ruimte "c" aanliggend moet zijn bij zowel ruimte "keuken" als ruimte "b" en ruimte "keuken", een lijst van punten waarop ruimte "c" kan liggen zodat ruimte "c" aanliggend is bij ruimte "keuken". We nemen ook een lijst van punten waarop ruimte "c" kan liggen zodat ruimte "c" aanliggend is bij ruimte "b". Als we deze twee lijsten als twee verzamelingen zien, is de doorsnede van deze verzamelingen de lijst van punten waarop ruimte "c" kan liggen zodat ruimte "c" bij ruimte "b" en bij ruimte "keuken" aanligt. Met andere

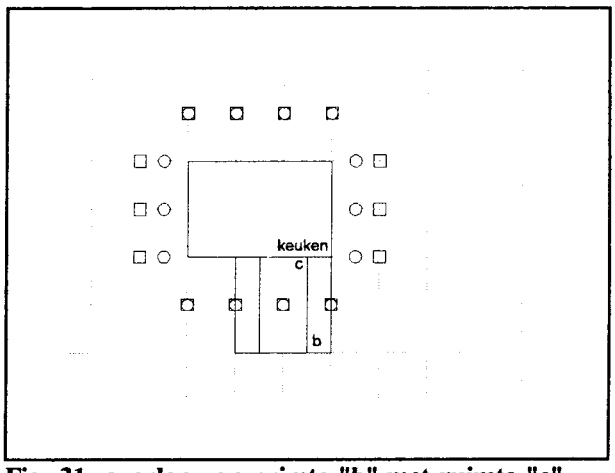


Fig. 31, overlap van ruimte "b" met ruimte "c"

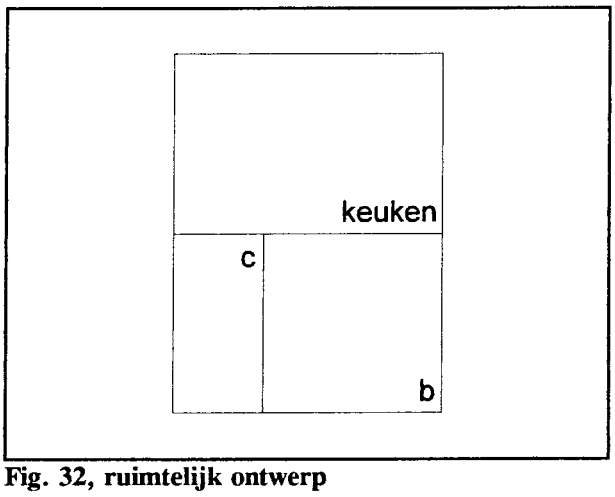


Fig. 32, ruimtelijk ontwerp

woorden moet een punt in beide lijsten aanwezig zijn om te zorgen dat ruimte "c" bij beide ruimten aanligt. Voor de duidelijkheid is figuur 30. Het kan goed zijn dat ruimte "c" ergens geplaatst wordt waar ruimte "c" ruimte "b" zal overlappen, ook dit is te zien en wel in figuur 31. Er zal dus ook op gelet moeten worden dat deze ruimte geen andere ruimten overlapt. Wordt een oplossing die voldoet gevonden, dan kan door gaan worden met het plaatsen van alle andere ruimten volgens dezelfde methode als hierboven beschreven voor ruimte "keuken", "b" en "c".

Nadat een ruimtelijk ontwerp van de eerste laag gemaakt is, vraagt het algoritme of het ontwerp meegenomen wordt naar de volgende ontwerpstap: het zoneren. Alle ontwerpen die de gebruiker kiest worden meegenomen. Verder kan de gebruiker het ontwerp nog weglaten of wijzigen door een ruimte van het ontwerp een nieuwe plaats te geven en dan meenemen naar het zoneren. Kort samengevat hebben we nu het volgende gedaan met de vetgedrukte invoer enkele alinea's terug:

1) **Neem een eerste willekeurige ruimte en plaats deze op positie (20,20) in het raster van 1 bij 1 meter.**

2) **Zoek in de lijst met voorwaarden een ruimte die samen met de geplaatste ruimte aanliggend moet zijn en stel een lijst samen van rasterpunten die voldoen, zodat de te plaatsen ruimte aanligt bij de ruimte op (20,20).**

3) **Neem een van deze punten uit de lijst en plaats de te plaatsen ruimte.**

4) **Neem een ruimte die moet aanliggen volgens de voorwaarden aan een van de geplaatste ruimten en maak lijsten (per geplaatste ruimte) van rasterpunten waar de te plaatsen ruimte kan liggen indien de ruimte moet aansluiten op een geplaatste ruimte.**

5) **Neem van al deze lijsten de doorsnede uit de verzamelingsleer en neem een rasterpunt van deze doorsnede. Plaats hier de ruimte als de ruimte geen al geplaatste ruimten overlapt. Neem anders een ander rasterpunt. Indien er geen rasterpunt gevonden kan worden is er geen oplossing.**

6) **Begin opnieuw bij punt 4 zolang er te plaatsen ruimten zijn.**

7) **Vraag of ontwerp nog gewijzigd moet worden, meegenomen wordt of weggelaten wordt.**

5.5 Zonering

Nu is dan een ruimtelijk ontwerp voor de eerste laag van het gebouw gereed. Dit ruimtelijk ontwerp zal nu gaan veranderen in een konstruktief ontwerp. Daarvoor is een hulpme-

thode ontwikkeld: de zogenaamde "zonering". In figuur 32 is een kant en klaar ruimtelijk ontwerp voor de eerste laag van het gebouw te zien, precies gemaakt volgens de methode die in de voorgaande alinea te zien was. Zones zijn, in dit afstudeerverslag, rechthoeken die 1 of meer ruimten kunnen bevatten. De zone (rechthoek) hoeft niet over het hele oppervlak bedekt te zijn met ruimten, maar langs de randen van de zone hoort wel altijd een ruimte te liggen, met ander woorden mag het niet mogelijk zijn een plek op de rand van de zone te ontdekken waar geen ruimte ligt. Voorbeelden van zones zijn te zien in figuur 33. Zones die niet voldoen zijn doorgestreept met een kruis. De zones worden aangegeven door een dikkere lijn dan de ruimten zelf. Behalve voorbeelden van een ruimtelijk ontwerp van "keuken", "b" en "c" zijn er ook twee andere ruimtelijke ontwerpen getekend om duidelijk aan te geven wat de definitie is van een zone. Er is zeer wel een discussie mogelijk over wat nu een zone moet zijn en wat niet, zeker omdat de keuze voor de definitie van een zone het konstruktief ontwerp beinvloedt: hier zou onderzoek naar gedaan kunnen worden. Hoe worden deze zones nu gevonden? Naast de invoer (vetgedrukt enkele alinea's hiervoor) is nu ook de positie van deze ruimten op het raster bekend. Arbitrair is gekozen voor een representatie van de positie van een ruimte door middel van het aangeven van de positie van het middelpunt van een ruimte op het raster. Aangezien de lengte en hoogte (hoogte betekende in dit afstudeerproject het verschil tussen twee y-waarden van punten uit het raster) van de ruimte bekend is uit de invoer, kunnen ook de posities van de hoekpunten van de ruimten berekenend worden. Bij voorbeeld kan gezegd worden dat het linkerhoekpunt van de ruimte "keuken" als volgt te bepalen is:

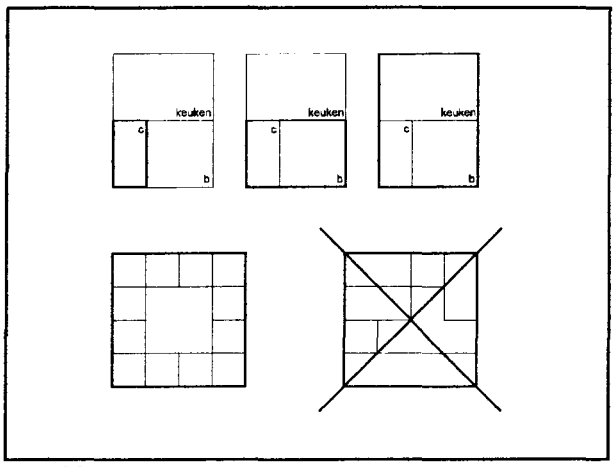


Fig. 33, voorbeelden van zones

Voor de x-waarde van het hoekpunt wordt de x-waarde van het middelpunt van de ruimte - de halve breedte van de ruimte genomen.

Voor de y-waarde van het hoekpunt wordt de y-waarde van het middelpunt van de ruimte + de halve hoogte van de ruimte genomen.

In figuur 34 is dit nog eens te zien.

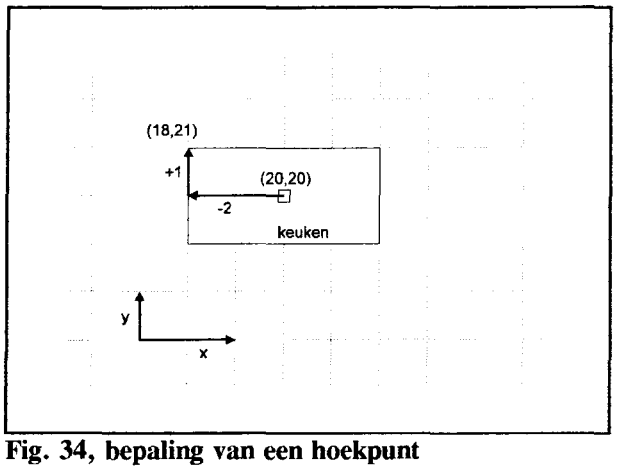


Fig. 34, bepaling van een hoekpunt

Bij een gegeven ruimtelijk ontwerp kunnen zo alle hoekpunten van alle ruimten bepaald worden. Dit is gedaan in figuur 35. De hoekpunten zijn aangegeven met kleine cirkels.

Als eenmaal alle hoekpunten van ruimten uit een ruimtelijk ontwerp gevonden zijn, kunnen ook zones gemaakt worden. Dit gebeurt als volgt (zie figuur 36) Er wordt een willekeurig hoekpunt genomen. Er wordt nu gezocht naar een hoekpunt met dezelfde y-waarde als het eerste hoekpunt en waarbij de x-waarde groter is dan de x-waarde van het eerste hoekpunt, zodat het eerste hoekpunt en het tweede hoekpunt op een horizontale lijn liggen en het tweede hoekpunt rechts naast het eerste hoekpunt ligt. Wordt zo'n hoekpunt gevonden, dan wordt gezocht naar een derde hoekpunt waarvan de x-waarde overeenkomt met de x-waarde van het tweede hoekpunt en waarvan de y-waarde kleiner is dan de y-waarde van het tweede hoekpunt, zodat het tweede hoekpunt en het derde hoekpunt op een verticale lijn liggen en het derde hoekpunt onder het tweede hoekpunt ligt. Nu wordt naar een vierde hoekpunt gezocht waarbij de y-waarde gelijk is aan de y-waarde van het derde punt en de x-waarde gelijk is aan de x-waarde van het eerste hoekpunt. Er wordt in deze richting gezocht (met de klok mee) om te voorkomen dat rechthoeken vaker gevonden worden en dus dubbele oplossingen verkregen worden. Wordt zo'n vierde hoekpunt gevonden dan is een rechthoek gevonden waarvan de hoekpunten bestaan uit een aantal hoekpunten uit de afbeelding "Hoekpunten van een ruimtelijk ontwerp".

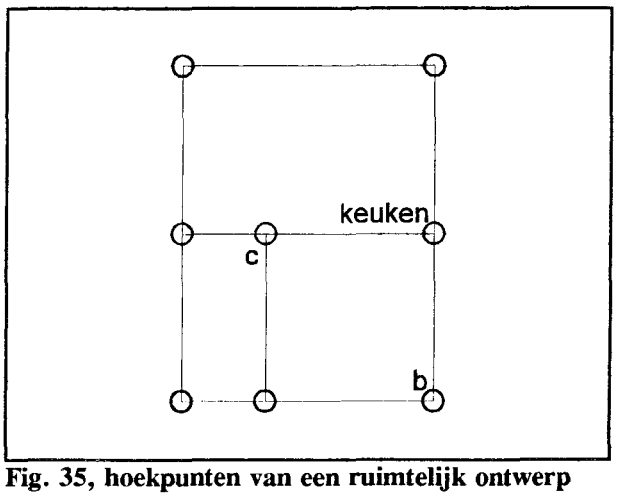


Fig. 35, hoekpunten van een ruimtelijk ontwerp

Kort samengevat wordt dus als volgt een rechthoek gevonden:

1) **Neem alle hoekpunten van alle ruimten in het ruimtelijk ontwerp.**

2) **Neem een hoekpunt en zoek in de richting van de klok naar andere hoekpunten zodat een rechthoek gevonden wordt. Dit kan door aan de x- en y-waarden van punten eisen te stellen.**

De gevonden rechthoeken hoeven echter nog geen zones te zijn zoals dat in dit afstudeerproject is afgesproken. In figuur 33 is te zien dat er best rechthoeken zijn die niet als "zone" worden aangemerkt, bijvoorbeeld omdat langs een gedeelte van de rechthoek geen zone ligt. Er zal dus onderzocht moeten worden of een rechthoek wel een zone is. Daartoe wordt de linkerbovenhoek van de rechthoek die we willen onderzoeken genomen. Vervolgens wordt bekeken van welke ruimte dit punt de linkerbovenhoek is. In figuur 37 is te zien dat ruimte "a" hieraan voldoet. Vervolgens wordt het rechterbovenhoekpunt van

ruimte "a" genomen en wordt gekeken of dit een linkerbovenhoekpunt van een andere ruimte is. Zo ja, dan sluit er een ruimte aan op ruimte "a" en is de rand van de rechthoek tot nu toe gesloten door ruimten. Zo nee, dan kan het rechterbovenhoekpunt van "a" nog het rechterbovenhoekpunt van de rechthoek zijn. Dit is in figuur 37 het geval bij ruimte "d". Het rechterbovenhoekpunt van een ruimte moet dus altijd of het linkerbovenhoekpunt van een andere ruimte zijn, of het rechterbovenhoekpunt zijn van de rechthoek. Ook hier wordt weer gezocht in de richting van de klok:

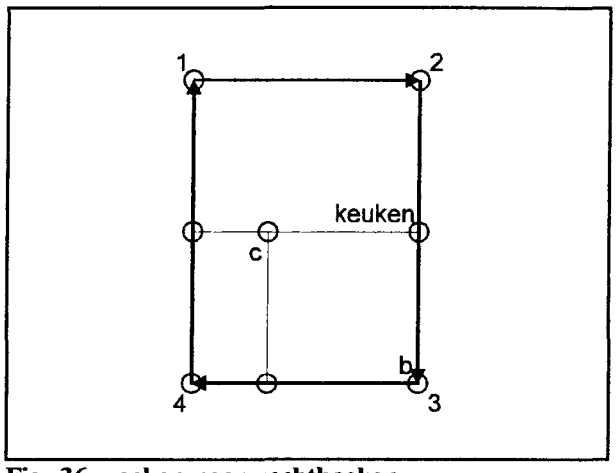


Fig. 36, zoeken naar rechthoeken

Als het rechterbovenhoekpunt van een ruimte het linkerbovenhoekpunt van een andere ruimte is, dan wordt de volgende ruimte bij de kop genomen en wordt het rechterbovenhoekpunt van deze ruimte gepakt om te kijken of het weer een linkerbovenhoekpunt van een volgende ruimte is, bijvoorbeeld ruimten "a" en "b" in figuur 37.

Als het rechterbovenhoekpunt van een ruimte het rechterbovenhoekpunt van de rechthoek is, dan wordt het rechteronderhoekpunt van deze ruimte genomen en wordt bekeken of dit punt het rechterbovenhoekpunt is van een volgende ruimte die onder deze ruimte ligt, bijvoorbeeld de ruimten "d" en "e" in figuur 37.

Nu wordt dus aan de rechter verticale zijde van de rechthoek gezocht en er wordt een soortgelijke zoekactie als net gehouden:

Als het rechteronderhoekpunt van een ruimte het rechterbovenhoekpunt is van een volgende ruimte, dan wordt deze volgende ruimte bij de kop genomen en wordt het rechteronderhoekpunt van deze volgende ruimte gepakt om te kijken of het weer een rechterbovenhoekpunt is van een volgende ruimte, bijvoorbeeld de ruimten "f" en "g" in figuur 37.

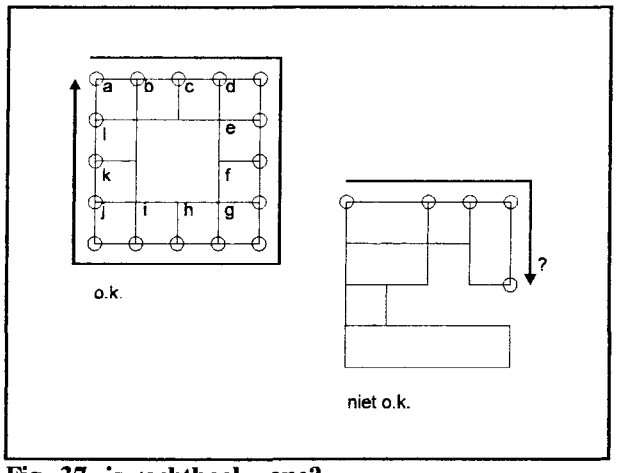


Fig. 37, is rechthoek zone?

ls het rechteronderhoekpunt van een ruimte het rechteronderhoekpunt van de rechthoek is, dan wordt het linkeronderhoekpunt van deze ruimte gepakt en wordt bekeken of dit punt

het rechteronderhoekpunt is van een volgende ruimte die links van deze ruimte ligt, bijvoorbeeld de ruimten "g" en "h" in figuur 37.

Nu wordt aan de horizontale onderzijde van de rechthoek gezocht en de zoekactie wordt vervolgd langs de rand van de rechthoek voor deze horizontale onderzijde en de linker verticale zijde van de rechthoek op een soortgelijke wijze als de zoekactie voor de 2 reeds behandelde zijden van de rechthoek. Wordt ontdekt dat het linkerbovenhoekpunt van ruimte "a" gelijk is aan het linkerbovenhoekpunt van de rechthoek in figuur 37, dan is dat een teken dat de rechthoek een zone is. Wordt ergens op de route langs de rand van de rechthoek een hoekpunt tegengekomen dat niet aan een van de twee voorwaarden voldoet (een hoekpunt van een andere ruimte of een hoekpunt van de rechthoek) dan is de rechthoek geen zone, zoals aangegeven in figuur 33.

Er is op de volgende manier gekeken of een rechthoek een zone is:

1) **Begin met de ruimte linksboven in de rechthoek en neem het hoekpunt van deze ruimte het verst in de zoekrichting langs de zoeklijn (deze zoeklijn is de pijl in de afbeelding "Is rechthoek zone?".**

2) **Dit punt moet het hoekpunt langs de zoeklijn van een andere ruimte zijn of het hoekpunt van de rechthoek.**

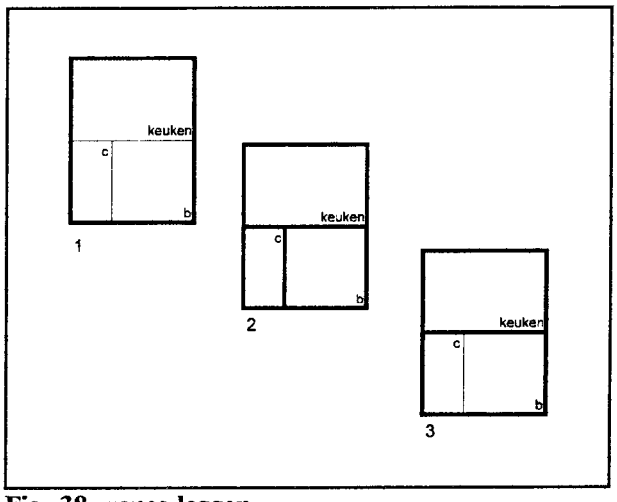


Fig. 38, zones leggen

3a) **Als het punt het hoekpunt is van een andere ruimte, ga dan langs de zoeklijn verder met zoeken.**

3b) **Als het punt hoekpunt is van de rechthoek ga dan in de volgende richting van de zoeklijn zoeken.**

4) **Een rechthoek is een zone als de zoeklijn op het einde sluit.**

Stel nu dat een ruimtelijk ontwerp is gevonden zoals dat te zien is in figuur 32 en dat volgens voorgaande regels alle zones zijn bepaald die het ruimtelijk ontwerp bevat. Een ruimtelijk ontwerp leidt tot meerdere zone-ontwerpen. (in het voorbeeld 3). Eenvoudig is in te zien, door het eenvoudige ruimtelijk ontwerp, dat deze zones de volgende zijn:

Opl. 1: De zone bestaande uit de keuken, ruimte "b" en ruimte "c".
Opl. 3: De zone bestaande uit de ruimten "b" en "c".
Opl. 2 + 3: De zone bestaande uit de keuken.
Opl. 2: De zone bestaande uit de ruimte "b".
Opl. 2: De zone bestaande uit de ruimte "c".

Het ruimtelijk ontwerp kan opgebouwd worden uit de zones. Dat wil zeggen dat het ruimtelijk ontwerp als ondergrond gebruikt wordt en de zones als puzzelstukjes. Er wordt geprobeerd met de zones, de puzzelstukjes, het ruimtelijk ontwerp "vol te leggen". Het resultaat daarvan is te zien in de figuur 38. Dit proces functioneert door een zone te nemen en te kijken welke ruimten deze zone bevat. Bevat de genomen zone niet alle ruimten van het ruimtelijk ontwerp dan wordt nog een zone (wel moet opgelet worden dat deze zone niet een ruimte bevat die al door een andere zone omvat is) genomen tot dat de verzameling zones alle ruimten van het ruimtelijk ontwerp bevat. Er is te zien in figuur 38 dat het ruimtelijk ontwerp op vele manieren gezoneerd kan worden. Alle mogelijke gezoneerde ontwerpen worden meegenomen naar het konstruktie-algoritme.

Kort samengevat wordt het ruimtelijk ontwerp dus als volgt gezoneerd:

1) **Alle zones in het ruimtelijk ontwerp worden gezocht.**

2) **Er wordt een configuratie van zones gezocht zodat elke ruimte van het ruimtelijk ontwerp in een zone vertegenwoordigd is, met die restrictie dat een ruimte niet in twee zones vertegenwoordigd is:**

3) **Er wordt een zone genomen en de ruimten die deze zone bevat worden onthouden.**

4) **Er wordt een volgende zone genomen en gekeken of deze volgende zone geen ruimten bevat die de al eerder genomen zones bevatten en de ruimten die deze volgende zone bevat worden onthouden.**

5) **Als alle ruimten van het ruimtelijk ontwerp in een laag in een zone gevat zijn kan gestopt worden.**

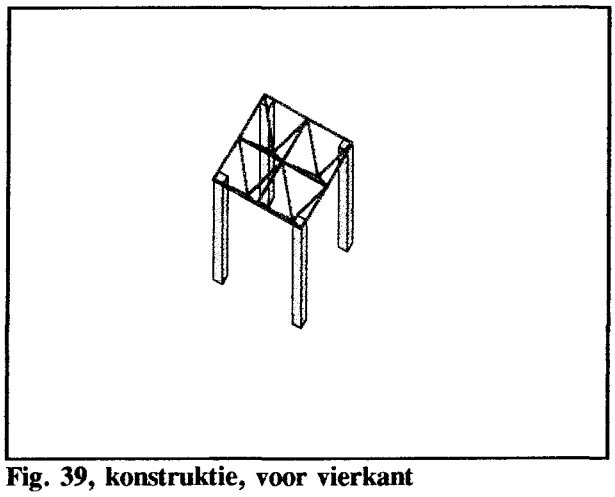


Fig. 39, konstruktie, voor vierkant

5.6 Konstrueren van zones

De zones uit voorgaande tekst worden apart gekonstrueerd, bijna alsof het zelf kleine gebouwtjes zijn, volgens regels die in de konstruktie-wereld gebruikt worden. De kennis hoe deze zones gekonstrueerd worden zijn per konstrukteur, per land en per situatie verschillend. Om een werkend algoritme te verkrijgen wordt heel eenvoudig begonnen en er wordt een zone uit het ruimtelijk ontwerp genomen. Er wordt volstaan met deze paar zones omdat later relatief gemakkelijk het algoritme uitgebreid kan worden met konstruktiemogelijkheden voor meer zones. Er worden nu de volgende zones beschouwd die het algoritme kan verwerken:

2 * 2 meter -> konstruktie 1, fig. 39,
4 * 4 meter -> konstruktie 1, fig. 39,
2 * 4 meter -> konstruktie 2, fig. 40 of konstruktie 3, fig. 41,
4 * 2 meter -> konstruktie 2, fig. 40 of konstruktie 3, fig. 41,
x * x meter -> konstruktie 4, fig. 42, alleen voor laag zonder bovenbouw.

Stel nu dat deze zone vierkant (2 * 2 of 4 * 4) is, zoals te zien is in figuur 39. Er kan op iedere hoek van de zone een kolom geplaatst worden en op deze vier kolommen een naar twee zijden dragend ruimtevakwerk. Dit ruimtevakwerk ligt geheel binnen de ruimte hetgeen wil zeggen dat de bovenzijde van het ruimtevakwerk de bovenzijde is van de ruimtelijke vorm. De kolommen worden arbitrair binnen de ruimtelijke referentie geplaatst, evengoed zouden ze buiten of midden in de ruimtelijke vorm geplaatst kunnen worden .Voor de dikte van de betonnen kolommen wordt 1/15 van de hoogte van de ruimte genomen. Voor de dikte van het ruimtelijk vakwerk wordt 1/25 van de overspanning genomen. Deze gegevens komen uit "The way we build now". Zijn de dikte van de kolommen en ruimtelijk vakwerk bekend, dan is het mogelijk om de stramienlijnen te bepalen door eenvoudige berekeningen.

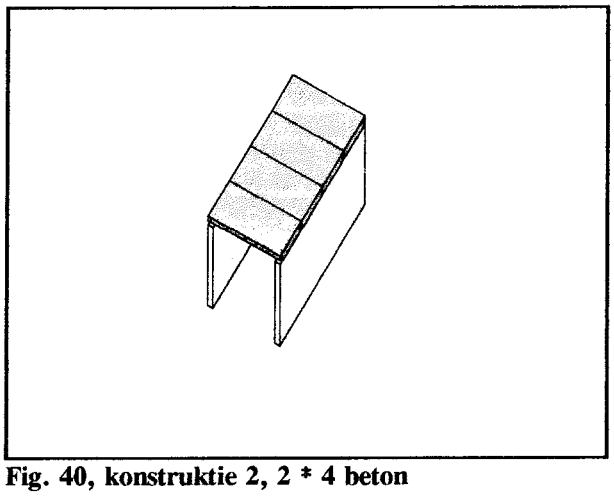


Fig. 40, konstruktie 2, 2 * 4 beton

<u>Konstruktie 1</u>:

- Vier betonnen kolommen op de hoeken, binnen ruimtelijke referentie, doorsnede 1/15 van de hoogte.
- Stalen ruim. vakwerk, dikte 1/25 van de hoogte, binnen ruimtelijke referentie.

Is de zone 2 * 4 of 4 * 2 meter, dan zijn in het algoritme voor deze zone 2 gekonstrueeerde oplossingen, te zien in de figuren 40 en 41.

Bij de eerste oplossing -in beton- zorgen twee betonnen schijven aan de lange zijden van de zone, dik 20 centimeter -bij metselwerk kan hiervoor bijvoorbeeld 105 mm. gebruikt worden- voor het dragen van de betonsysteemplaten -in dit geval 1 meter breed- met een dikte van 1/20 van de overspanning. De betonnen schijven worden arbitrair binnen de ruimtelijk vorm geplaatst zoals de betonsysteemplaten bovenop de ruimtelijke vorm geplaats worden.

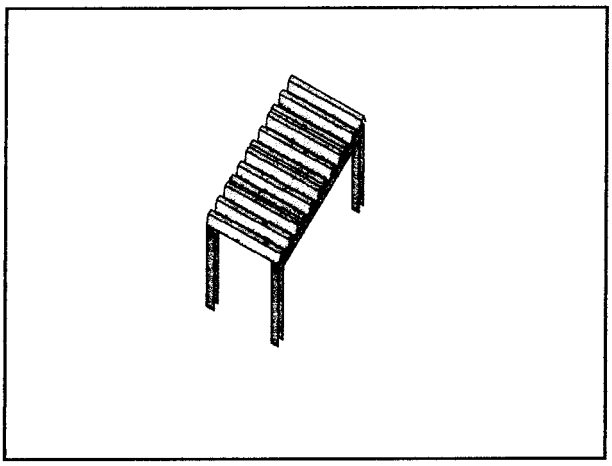


Fig. 41, konstruktie 3, 2 * 4 staal

Konstruktie 2:

- Twee schijven beton, 20 cm. dik aan langste zijden, binnen ruim. referentie.
- Betonnen vloerelementen, 1/20 van de overspanning dik op schijven.

Bij de tweede oplossing -in staal- zorgen aan elke zijde van de zone twee stalen kolommen, 1/15 van de hoogte, en een stalen balk met een hoogte van 1/20 van de overspanning voor het dragen van de geprofileerde staalplaat met een dikte van 1/25 van de overspanning. Het raamwerk van kolommen en balken is binnen de ruimtelijk vorm geplaatst, de geprofileerde staalplaat bovenop de ruimtelijk vorm. Evengoed zijn andere plaatsingsvormen van konstruktie-elementen in het programma te verwerken.

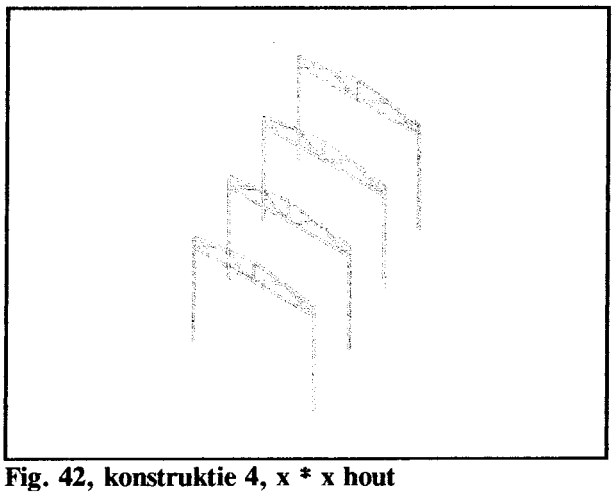


Fig. 42, konstruktie 4, x * x hout

Konstruktie 3:

- Twee kolommen, 1/15 van de hoogte in doorsnede met balk, 1/20 van de overspanning hoog, tezamen vormend een raamwerk langs de twee lange zijden binnen de ruimtelijke referentie.
- Stalen geprofileerde vloerplaten, omega-profiel, liggend aan weerszijden op het raamwerk, 1/25 van de overspanning hoog.

Omdat alle konstruktie-elementen bij deze zones globaal gedimensioneerd worden, is het mogelijk de stramienlijnen wederom te bepalen. Bovendien kan het gewicht aan staal en beton redelijk bepaald worden.

Konstruktie 4:

- Aan kopse zijden van ruimte en verder om de twee meter h.o.h. houten portalen bestaande uit twee houten kolommen en 8 vakwerkliggers, onafhankelijk van de afmetingen van de zone.

Deze mogelijkheden zijn beslist niet voldoende voor een goed konstruktief ontwerp, omdat:

- De stabiliteit van de konstrukties 1 t/m 4 alsmede de stabiliteit van de samenstellingen van de konstrukties 1 t/m 4 in een gezoneerd ruimtelijk ontwerp wordt niet in beschouwing genomen.

- Bij combinaties van konstrukties 1 t/m 4 ontstaan moeilijk uitvoerbare gebouwen en tevens moeilijke verbindingen tussen de konstrukties.

- Wanneer twee konstrukties naast elkaar liggen en kolommen van beide konstrukties vrijwel naast elkaar staan is vaak een samenstelling mogelijk waardoor er slechts 1 kolom nodig is. Hoewel ten dele als gevolg van het zoneringsprincipe, is dit ook een nadeel van de konstruktievarianten.

- Er zijn veel te weinig konstrukties (slechts 4) waardoor geen realistisch konstruktief ontwerp wordt gegenereerd.

Ze zijn slechts bedoeld om een werkend programma te verkrijgen, opdat algoritmen en ideeen uitgeprobeerd kunnen worden op werkzaamheid. Deze beperkte hoeveelheid konstruktieve uitwerkingen van zones is inderdaad beperkt tot een aantal zones. Wel is het zo dat de zones een variabele hoogte kunnen krijgen en dat erg gemakkelijk andere dimensioneringsregels kunnen worden ingevoerd.

Stel nu dat er een gezoneerd ruimtelijk ontwerp is, bijvoorbeeld zoals dat te zien is in figuur 38. Zones in deze afbeelding worden met dikke lijnen aangegeven. De ruimten b is 4 * 4, c is 2 * 4 meter en de keuken is 6 * 4 meter. Volgens bovenstaande kennis worden de gezoneerde varianten als volgt gekonstrueerd:

Variant 1: zone 6 * 8 (keuken, c en b): konstruktie 4, fig. 42

Variant 2:	zone 2 * 4 (c):	konstruktie 2 ,	fig. 40
	zone 4 * 4 (b):	konstruktie 1,	fig. 39
	zone 6 * 4 (keuken):	konstruktie 4,	fig. 42
Variant 2:	zone 2 * 4 (c):	konstruktie 3,	fig. 41
	zone 4 * 4 (b):	konstruktie 1,	fig. 39
	zone 6 * 4 (keuken):	konstruktie 4,	fig. 42
Variant 3:	zone 6 * 4 (c en b):	konstruktie 4,	fig. 42
	zone 6 * 4 (keuken):	konstruktie 4,	fig. 42

Voor 1 ruimtelijk ontwerp van 1 laag worden zo 4 mogelijkheden om dit ruimtelijk ontwerp te konstrueren gevonden. Vervolgens wordt uitgerekend hoeveel staal en beton bij het ontwerp gebruikt worden door de hoeveelheden staal en beton per zone gebruikt te sommeren. Het algoritme presenteert het ontwerp met de hoeveelheden gebruikt staal en beton en laat het aan de gebruiker over of dit ontwerp meegenomen wordt naar de volgende ontwerpstap. Zo worden alle ontwerpen behandeld. Alle stappen tot nu toe:

1) **Er wordt een gezoneerd ruimtelijk ontwerp genomen**

2) **Per type of grootte zone zijn een aantal manieren om deze zone te konstrueren bekend, dit zijn "heuristics": kennisregels. Tot nu toe zijn slechts een paar kennisregels toegepast.***

3) **Alle mogelijkheden om elke zone van het gezoneerde ruimtelijk ontwerp te konstrueren worden nagegaan. Hierdoor ontstaan er meedere gekonstrueerde gezoneerde ruimtelijke ontwerpen.**

4) **De massa gebruikt beton en staal worden uitgerekend. De gebruiker kan kiezen om de gekonstrueerde variant mee te nemen of niet naar de volgende ontwerpstap.**

* **Deze kennisregels zijn in voorgaande tekst besproken, de kennis om de in figuur 39 t/m 42 getoonde konstrukties te maken.**

5.7 Selectiecriteria

Als op deze manier gewerkt wordt om een compleet konstruktief ontwerp te laten genereren zien we dat er bijna oneindig veel goede mogelijkheden onstaan voor het konstruktief ontwerp. Deze gegenereerde mogelijkheden zijn globaal beschouwd ook goed. Er zullen geen vreemde ontwerpen gegenereerd worden die bijvoorbeeld fysisch nauwelijks mogelijk zijn. Om dit te illustreren kan nog eens gekeken worden naar figuur 38 en het daarbij behorende verhaal. Er is te zien dat al het ruimtelijk ontwerp van de eerste laag een hoop mogelijkheden geeft indien deze laag gekonstrueerd wordt. Bij het plaatsen van ruimten van de tweede laag op deze eerste laag zullen de mogelijkheden alleen nog maar toennemen. Het blijkt nodig te zijn een aantal van de gegenereerde ontwerpen te selecteren om vervolgens met deze ontwerpen door te gaan. Het is beter om interactief met het algoritme te kunnen werken, dat wil zeggen dat tijdens het genereren van een konstruktief ontwerp, de gebruiker van het algoritme moet kunnen ingrijpen om het ontwerpproces te veranderen.

Uit de vele konstruktieve ontwerpen die het algoritme uit het vorige hoofdstuk maakt, moeten enkele ontwerpen gekozen worden als het algoritme zover is dat de eerste laag geheel konstruktief ontworpen is. Dit omdat de benodigde rekentijd aanzienlijk wordt, wanneer zonder meer alle konstruktieve ontwerpen van de eerste laag gebruikt zouden worden bij het plaatsen van ruimten op de eerste laag. Maar als deze konstruktieve ontwerpen nu bekeken worden, zien we dan welke ontwerpen we zullen nemen en welke ontwerpen reeds zullen wegvallen?

Het lijkt verstandig een aantal eigenschappen van de konstruktieve ontwerpen van de eerste laag te bepalen. De gebruiker kan vervolgens aan de hand van deze eigenschappen bepalen welke ontwerpen meegenomen worden naar de volgende ontwerpstap. Een aantal mogelijke eigenschappen van de konstruktieve ontwerpen.

1) Hoeveelheid gebruikt materiaal per soort materiaal, bijvoorbeeld een hoeveelheid staal, een hoeveelheid beton, enz. Hier zijn wel globale dimensioneringsberekeningen voor nodig.

- Aantal zijden van een ruimte die aan de buitenkant liggen, met andere woorden: de zijden waarbij, indien er een ramen in aangebracht worden, direct buitenlicht naar binnen kan stralen. Dit is onder andere een bouwfysica-aspect.

- Aantal gebruikte konstruktieve hoofddraagvarianten. Is het ontwerp geheel opgebouwd volgens het principe van schijven, balken en kolommen of kolommen en ruimtelijke vakwerken? Of bestaat het ontwerp uit meerdere konstruktieve principes? Dit is onder andere een konstruktief-ontwerp-aspect.

- Het gemak waarmee, uitgaande van deze huidige laag, er straks ruimten op deze laag geplaatst kunnen worden. Hoeveel mogelijkheden uit de afbeelding "Plaatsingsmogelijkheden zijn mogelijk? Dit is een konstruktief aspect.

- Totale loopafstand van alle wegen tussen alle ruimten. Dit is een graadmeter voor de "compactheid" van het ontwerp. Dit is een ruimtelijk-ontwerp-aspect.

- De ligging van staalkonstrukties ten opzichte van schijven. Dit in verband met de uitvoeringsvolgorde tijdens het bouwen. Dit is een uitvoeringsaspect.

- De verhouding buitenoppervlak/volume. Dit is onder andere een bouwfysica-aspect.

In dit afstudeerproject wordt de hoeveelheid beton en staal bepaald door de globaal gedimensioneerde konstruktie-elementen te nemen en deze hoeveelheid te berekenen, per laag en onafhankelijk van andere lagen, precies zoals beschreven is in de bijlage "Gebruiksaanwijzing, code en toelichting bij het programma". Dit wordt methode 1a genoemd. Er is echter nog een andere methode mogelijk die hieronder beschreven wordt als methode 1b. Deze methode is niet toegepast in het afstudeerproject. Alle andere criteria worden hier slechts genoemd en niet verder uitgewerkt in dit afstudeerproject.

1b) Hoeveelheid gebruikt materiaal per soort materiaal (niet toegepaste methode)

Hiervoor is ook kennis nodig omtrent het dimensioneren van konstrukties, afkomstig uit het vakgebied konstruktief ontwerpen. Stel dat een ruimtelijk ontwerp van de eerste laag aanwezig is. Nu kan globaal bepaald worden hoeveel materiaal nodig is voor de draagkonstruktie. Omdat niet bekend is wat nog allemaal bovenop deze eerste laag geplaatst wordt (misschien wel 30 verdiepingen!) kan niet meer gedaan worden dan er vanuit gaan dat deze eerste laag de laatste laag is. Er wordt dan een gewone dakbelasting op deze eerste laag geplaatst en gekeken hoeveel materiaal ervoor nodig is om deze laag de dakbelasting af te laten dragen. Het is de vraag of de variant die het gunstigste de dakbelasting afleid naar de grond (dus met het minste materiaal) ook wel de variant is die het gunstigste de belasting afdraagt van misschien wel 30 verdiepingen op deze eerste laag. Er wordt voor deze belasting 1,0 kN/m^2 genomen. Dit is te zien in figuur 43. Vervolgens wordt per gekonstrueerde zone deze

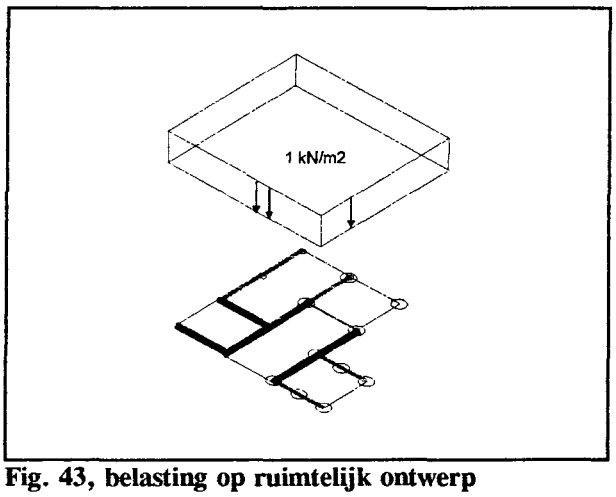


Fig. 43, belasting op ruimtelijk ontwerp

belasting bekeken. Dit is te zien in figuur 44. Vervolgens kunnen per zone de elementen die in deze zone aanwezig zijn berekend worden met de daartoe geschikte rekenregels.

Al deze informatie, betreffende hoe de hoeveelheid materiaal te bepalen in een konstruktief ontwerp gemaakt door een algoritme, door een globale dimensionering van de konstruktieve elementen, moet met de nodige voorzichtigheid bekeken worden. De informatie tot nu toe handelt slecht over 1 laag van ruimtelijke elementen. Zoals reeds eerder gezegd, is het maar zeer de vraag hoe zwaar konstruktieve elementen op de eerste laag moeten zijn als er nog ruimten op deze eerste laag geplaatst worden. Het is in dit verband onmogelijk hier antwoord op te geven. Het is wel mogelijk een suggestie te geven hoe bij een ontwerp van meer verdiepingen de konstruktieve elementen gedimensioneerd zouden kunnen worden, zie hiervoor onder andere figuur 45. In dit figuur is te zien dat op de eerste laag een grote vierkante ruimte ligt. Op een deel van het bovenvlak van deze ruimte, is een rechthoekige ruimte geplaatst op de tweede laag, ongeveer met de halve grootte van de ruimte op de eerste laag. Op de ruimte op de tweede laag, is een kleine vierkante ruimte geplaatst. Dit is de derde laag. Stel nu dat op alle dakvlakken een uniforme dakbelasting rust en op alle vloeren in alle ruimten een vloerbelasting. Er kan dan begonnen met de ruimte op de derde laag. Een konstruktietype voor deze ruimte wordt gekozen (in dit verband zouden de genoemde ruimten ook best zones kunnen zijn, vandaar dat gezegd wordt dat een konstruktietype gekozen wordt voor de ruimte, normaal worden konstruktietypen gekozen voor zones), en wordt de dakbelasting van de ruimte op de derde laag in beschouwing genomen. Zoals dat eerst al te zien was bij de gewichtsbepaling van konstruktievarianten van zones, kunnen nu de konstruktie op de derde laag gedimensioneerd worden. Omdat het konstruktie-type bekend is, de dimensies van de konstruktie, de vloerbelasting van de ruimte op de derde laag en de krachten op de konstruktie-elementen van de derde laag, kan bepaald worden welke krachten, behalve de dakbelasting van sommige ruimten, op de

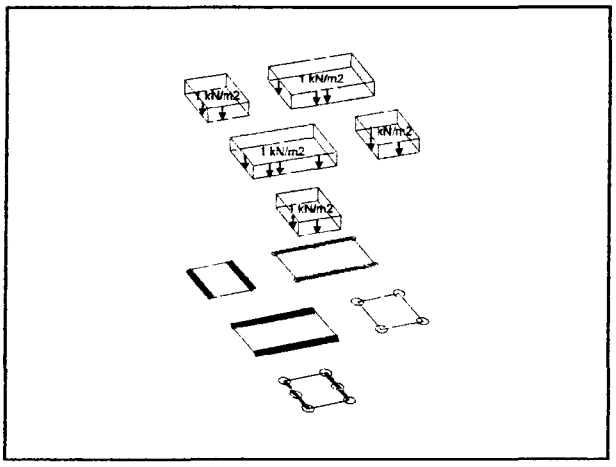


Fig. 44, belasting per zone

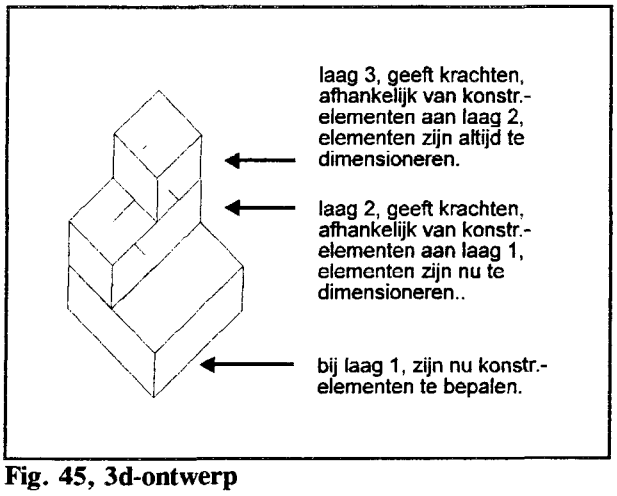


Fig. 45, 3d-ontwerp

tweede laag inwerken. Er wordt voor de tweede laag per zone een konstruktietype gekozen en vervolgens kan dezelfde methode gebruikt worden als voorheen beschreven om de konstruktie te dimensioneren van de tweede laag. Uiteindelijk wordt de beschreven methode nogmaals gebruikt om de konstruktie van de eerste laag globaal te dimensioneren. Kortom:

1) **Begin bij de bovenste laag van een ontwerp. Kies een konstruktietype per zone van de derde laag.**

2) **Verwerk de dakbelasting en dimensioneer de konstruktie van de bovenste laag, zoals dat reeds beschreven is bij een voorbeeld met een ruimtelijk ontwerp van 1 laag.**

3) **De massa van de konstruktie op de bovenste laag is nu bekend. Neem een konstruktietype per zone voor de onderliggende laag, dit noemen we de beschouwde laag. Nu is bekend hoe de vloerbelastingen en konstruktie-element-belastingen (krachten en eigen gewicht) van de laag boven de beschouwde laag afgedragen worden. Neem eventuele dakbelasting mee op de beschouwde laag.**

4) **Dimensioneer de konstruktie-elementen op de beschouwde laag.**

5) **De massa van de konstruktie op de beschouwde laag is nu bekend. Neem een konstruktietype per zone voor de onderliggende laag, dit noemen we nu de beschouwde laag. Nu is bekend hoe de vloerbelastingen en konstruktie-element-belastingen (krachten en eigen gewicht) van de laag boven de beschouwde laag afgedragen worden. Neem eventuele dakbelasting mee op de beschouwde laag.**

6) **Ga naar punt 4 tot de onderste laag. De konstruktie is nu globaal gedimensioneerd.**

5.8 Ruimtelijk ontwerp met voorwaarden

Nu is de eerste laag ruimten geordend, gezoneerd en gekonstrueerd. De hoofdlijnen van deze acties zijn dikgedrukt in de voorgaande tekst te lezen. Nu wordt de tweede laag ruimten op de eerste laag ruimten geplaatst. Is er geen tweede laag dan is het algoritme beeindigd. Omdat telkens een gekonstrueerd, gezoneerd ruimtelijk ontwerp genomen wordt om hierop de ruimten van de tweede laag te plaatsen is er reeds enige informatie over de konstruktie-elementen aanwezig. Uit de invoer, die enkele bladzijden terug vermeld staat, wordt een willekeurige ruimte die op de tweede laag geplaatst moet worden genomen. Deze ruimte wordt niet op een willekeurige manier geplaatst. Een ruimte, hoewel meerdere ruimten soms door het zoneringsprincipe als 1 grote ruimte gezien worden, is toch altijd konstruktief een soort kleinste element. Zo'n ruimte mag niet zomaar willekeurig op de onderliggende laag, die we net gemaakt hebben, geplaatst worden. De ruimten op de

tweede laag moeten vrij gemakkelijk gedragen kunnen worden door de konstruktie-elementen van de eerste laag. In figuur 46 is te zien wanneer een ruimte wel op de onderliggende laag geplaatst kan worden. In figuur 46 zijn de zichtbare konstruktie-elementen onderliggend ten opzichte van de nu nog te plaatsen ruimten! Als onder een ruimte, in de eerste laag, de konstruktie-elementen zo verdeeld zijn als op 1 van de goede mogelijkheden in figuur 46, dan kan de ruimte geplaatst worden op deze konstruktie-elementen. Let wel, de in de afbeelding vereiste konstruktie-elementen zijn de minimaal benodigde elementen. Mochten er onder de ruimte, of langs de ruimtelijnen nog extra konstruktie-elementen liggen, dat is dat alleen maar beter in veel gevallen. Het programma-algoritme wat betreft space-allocation werkt nu dan ook wat anders. Als namelijk bekend is bij welke voorwaarden een ruimte op een laag met konstruktie-elementen geplaatst kan worden, zie de afbeelding, dan kan ook gezorgd worden dat we de te plaatsen ruimte niet willekeurig geplaatst wordt om daarna te kijken of aan de plaatsingvoorwaarden zoals in de afbeelding te zien is, is voldaan. De plaatsing van de ruimte is te reduceren door de volgende regels:

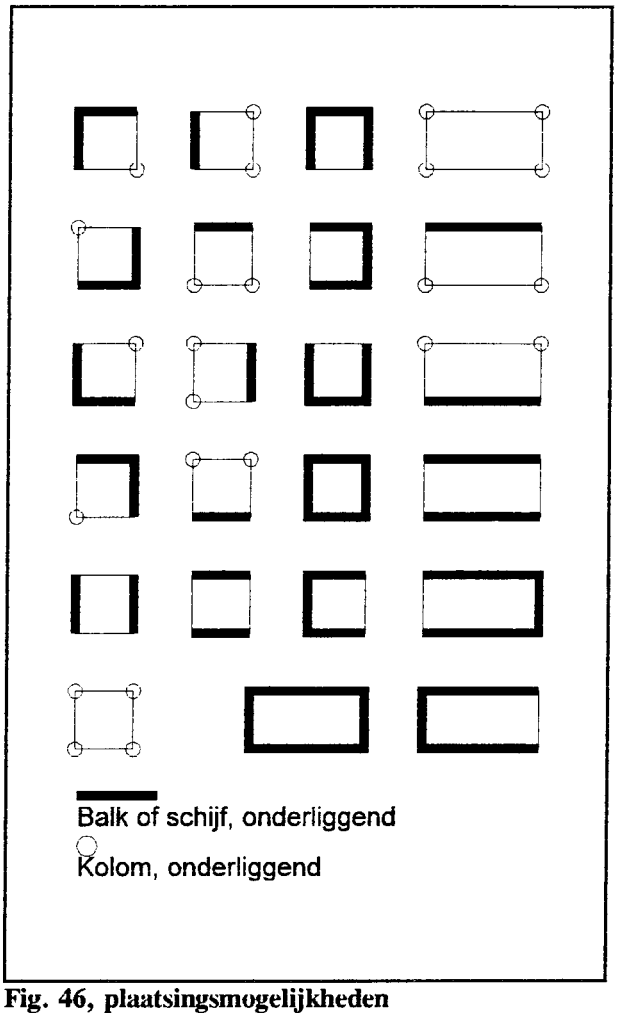


Fig. 46, plaatsingsmogelijkheden

1) **De linkerbovenhoek van de ruimte wordt geplaatst op een kolom of op een schijf of balk.**

2) **Er wordt gekeken of de ruimte wel geheel binnen een zone valt uit de vorige laag, Voorlopig wordt er vanuit gegaan dat dit een methode is om ervoor te zorgen dat konstruktief en ruimtelijk zinvolle ontwerpen onstaan.**

3) **Er wordt vastgesteld of de te plaatsen ruimte vierkant, dan wel rechthoekig is.**

4) **Is de te plaatsen ruimte vierkant, dan wordt getoetst we of:**

- **het rechterbovenhoekpunt, het rechteronderhoekpunt en het linkeronderhoekpunt van de te plaatsen ruimte op een kolom of op een schijf of balk staan. Elk hoekpunt mag gerust op een andere schijf staan, ook op de rand van zo'n schijf of balk.**

5) **Is de te plaatsen ruimte rechthoekig, dan wordt getoetst of:**

- **het linkerbovenhoekpunt en het rechterbovenhoekpunt op 1 schijf of balk staan. Verder of het rechteronderhoekpunt en het linkeronderhoekpunt op een kolom staan, of:**

- **het linkeronderhoekpunt en het rechteronderhoekpunt op 1 schijf of balk staan. Verder of het rechterbovenhoekpunt en het linkerbovenhoekpunt op een kolom staan, of;**

- **het linkeronderhoekpunt en het rechteronderhoekpunt op 1 schijf of balk staan en dat het linkerbovenhoekpunt en het rechterbovenhoekpunt op 1 schijf of balk staan, of:**

- **dat elk hoekpunt op een kolom staat.**

6) **Mocht de te plaatsen ruimte nu geen al reeds geplaatste ruimte overlappen en mocht er verder voldaan worden aan de voorwaarden wat betreft de aansluiting op andere ruimten van de te plaatsen ruimte, dan kan de ruimte geplaatst worden, anders moet er terug gegaan worden naar punt 1. Verder moet de ruimte binnen een zone van de vorige laag vallen zodat uitkragingen nu niet mogelijk zijn.**

Er is te zien bij de punten 3,4 en 5 dat precies alle mogelijke plaatsingen volgens figuur 46 gebruikt kunnen worden, zonder al deze plaatsingen ook werkelijk te controleren.

Op bovenstaande wijze worden ook de andere ruimten die op de tweede laag geplaatst dienen te worden geplaatst, tot het moment dat alle ruimten geplaatst zijn. Het is mogelijk dat op de tweede laag, (verdieping werd ook wel laag genoemd) nu een ruimtelijk ontwerp is ontstaan zoals in figuur 32. Weliswaar is in dit figuur het ruimtelijk ontwerp van de eerste laag te zien, maar een dergelijke voorstelling zou ook gemaakt kunnen worden van de tweede laag. Nadat het algoritme wederom het mogelijk maakt om dit ontwerp te veranderen of mee te nemen naar de volgende ontwerpstap, kan verder gegaan worden met dit ruimtelijk ontwerp van de tweede laag, door dit ontwerp te gaan zoneren volgens de regels die genoemd zijn. Ook kan hierna het gezoneerde ontwerp weer gekonstrueerd worden. Mocht er een derde laag zijn, dan kan deze laag hierna op de tweede inmiddels gekonstrueerde laag geplaatst worden, volgens de regels die reeds voor de tweede laag genoemd zijn. Samengevat doen we dus het volgende, ook te zien in figuur 26:

0) Invoer

1) Maak ruimtelijk ontwerp laag.

1a) Maak wijzigingen ontwerp of laat ontwerp wel of niet doorgaan naar 2.

2) Zoneer laag.

3) Konstrueer laag en geef aanvullende gegevens.

3a) Laat ontwerp wel of niet doorgaan naar 4.

4) Plaats ruimten volgende laag.

5) Ga naar punt 1a en herhaal deze cyclus voor elke laag tot er geen te plaatsen lagen meer zijn.

Er is gekozen voor het principe om van beneden naar boven te ontwerpen en te konstrueren. In principe is het mogelijk om met een andere laag dan de onderste te beginnen en andere lagen hierop aan te passen en te ontwerpen.

Bij 5.6 is te zien dat de konstruktie gekozen wordt bij een ruimteclustering, een zone. De konstruktie hangt dus af van het gedeeltelijk gereed ruimtelijk ontwerp. Bij deze paragraaf is te zien dat de ruimteplaatsing afhankelijk is van de konstruktie op de onderliggende laag. Het ruimtelijk ontwerp hangt dus af van het gedeeltelijk gereed konstruktief ontwerp.

De konstruktie op een laag hangt af van de konstruktie op de voorgaande laag. Ruimten op de behandelde laag worden immers geplaatst afhankelijk van de konstruktie van de voorgaande laag. Door de specifieke plaatsing van de ruimten zijn ook bepaalde zoneringen van toepassing en hierdoor zullen de zones op een bepaalde manier gekonstrueerd worden.

6 Een werkend programma

6.1 Inleiding

Zoals besproken bij het hoofdstuk Inleiding en doelstellingen is het van belang om de algoritmen te testen door ze te implementeren. Zo zijn de algoritmen getoetst op uitvoerbaarheid en kan gekeken worden of de algoritmen ook zinvol werken.

6.2 Declaratief versus procedureel

Uit het boek "Knowledge Systems and Prolog" halen we de volgende kennis [5]:

Er bestaan declaratieve en procedurele programmeertalen. Bij een procedurele taal wordt een aantal stappen beschreven om een probleem op te lossen. Bij een declaratieve taal worden regels en feiten genoemd die relaties beschrijven. Door deze declaratieve wijze van programmeren zijn stroomdiagrammen eigenlijk niet bruikbaar. Ook algoritmen zoals deze gebruikt zijn in de voorgaande hoofdstukken zijn niet zonder meer in een declaratieve taal te beschrijven. Veel problemen die opgelost kunnen worden met kunstmatige-intelligentie-algoritmen kunnen makkelijker in een declaratieve taal worden opgelost dan in een procedurele taal.

De regels die in een declaratieve programmeertaal gebruikt worden, zullen uitgevoerd worden door de computer. Uiteindelijk zal er dus wel terdege een vaststaande procedure moeten ontstaan uit een programma die een microprocessor kan sturen. Hiervoor zorgt de interference-engine, een programma dat de regels en feiten uit het declaratieve programma verwerkt zodat een computer dit programma kan uitvoeren. Een declaratieve programmeertaal heeft geen:

- Opdrachten die een variabele een waarde geven of een waarde veranderen
- Go-to opdrachten
- If-then-else opdrachten
- Do-loops, for-loops, en while-loops

Als wordt ervaren wat een declaratieve taal niet heeft, is het duidelijk dat stroomdiagrammen nauwelijks te implementeren zijn. Veel zaken die een declaratieve taal heeft en vaak ontbreken bij procedurele talen zijn:

- Predicaten die relaties beschrijven tussen entiteiten.

- Een methode om predicaten te maken door regels en feiten in het programma te voeren.

- Een methode om vragen te stellen en onderzoeken te starten door een doel te stellen aan het programma.

- Datastructuren die arrays kunnen simuleren van bestaande procedurele talen, alleen vaak universeler. Een array laat vaak maar 1 type variabele toe, deze datastructuren laten meerdere datatypen toe in 1 structuur.

- Een patroonherkenner die de datastructuren bouwt en onderzoekt.

- Ingebouwde predicaten voor rekenwerk, input/output en systeem-oproepen.

Met elke goede programmeertaal is het mogelijk alle mogelijkheden van een computer te benutten. Bij kunstmatige-intelligentie-problemen is het vaak gemakkelijker te programmeren in een declaratieve taal, in plaats van in een procedurele taal. Hieronder zien we in een fictieve procedurele taal hoe de faculteit van een getal wordt uitgerekend. Er wordt verteld hoe het programma het probleem moet oplossen. Eerst wordt het getal op een of andere manier ingevoerd, hier aangegeven met "Invoer A". Vervolgens wordt een variabele P op de waarde "1" gesteld door de opdracht "P=1". Met een "for"-loop wordt nu de variabele "P" telkens met de waarde van de variabele "Teller" vermenigvuldigd totdat de variabele "Teller" dezelfde waarde heeft als het ingevoerde getal "A". Nu heeft de variabele "P" dezelfde waarde als de faculteit van de waarde "A" en kan de waarde "P" uitgevoerd worden, hier symbolisch aangegeven met de zin "Uitvoer P".

```
Invoer A

P = 1

For Teller = 1 to A

        Begin
        P = P * Teller
        End

Uitvoer P
```

Nu wordt hetzelfde probleem geprogrammeerd in een declaratieve taal. Stel dat de faculteit van het getal "A" wordt gevraagd. Het probleem wordt declaratief beschreven door te stellen dat de faculteit van "A" eigenlijk de faculteit is van het getal "A-1" vermenigvuldigd met het getal "A", een recurrente betrekking. Verder is afgesproken dat de faculteit van 0 gelijk is aan 1. In een declaratieve taal wordt het probleem dan als volgt beschreven:

```
fac( 0, 1 ).
```

fac(N, X) ← fac(N-1, Y) & X is N * Y.

Het predicaat "fac" heeft 2 variabelen. De eerste variabele geeft het getal weer waar de faculteit van gevraagd is. De tweede variabele geeft de faculteit van dit getal. Het eerste predicaat "fac" geeft aan dat de faculteit van 0 de uitkomst 1 heeft. In de tweede programmaregel wordt verteld dat de faculteit van "N" te krijgen is door de faculteit van "N-1" te bekijken met als uitkomst de waarde "Y". De waarde "X", de uitkomst van de faculteit van "N", krijgen we door de waarde "N" met de waarde "Y" te vermenigvuldigen, dit gebeurt door het gedeelte "X is N*Y".

6.3 Waarom declaratief?

In het voorgaande hoofdstuk werd telkens over algoritmen gesproken. Nu duidelijk is dat algoritmen moeilijk te verwerken zijn in een declaratieve taal, komt de vraag naar boven waarom deze algoritmen in dit afstudeerproject in een declaratieve taal beschreven worden. Hier zijn een aantal redenen voor (er zijn al enkele redenen in 5.2 genoemd):

- De algoritmen lijken globaal gezien algoritmen. Wanneer deze algoritmen moeten worden uitgewerkt, zijn details vaak beter declaratief te verwerken. Wanneer bijvoorbeeld duidelijk gemaakt moet worden wat het hoekpunt van een zone is of wanneer een ruimte nog op een balk of schijf ligt.

- Zelfs op het globale vlak bezien zijn de algoritmen slechts ten dele algoritmen omdat niet 1 oplossing gevraagd wordt. Telkens worden alle ruimtelijke oplossingen gevraagd en van elke ruimtelijke oplossing weer alle zoneringsmogelijkheden en van deze zoneringsmogelijkheden per mogelijkheid weer alle konstruktiemogelijkheden. Binnen deze zoekmogelijkheden wordt ook nog selectief gesnoeid en veranderd. Dan is een programma met zogenaamde backtracking, dit is een zoekmechanisme, zie de bijlage "T8-T9-verslagen", ideaal. Backtracking is vaak aanwezig bij declaratieve programmeertalen en in ieder geval nooit bij procedurele talen.

- Bij het bouwen van kennissystemen en het verwerken van menselijke kennis door de computer is het in het algemeen gemakkelijker om dit met een declaratieve programmeertaal te doen.

- Het gebruik van een declaratieve programmeertaal heeft het grote voordeel dat eventueel later toegepaste kunstmatige-intelligentie-technieken gemakkelijk te implementeren zijn.

Er zijn ook nadelen aan het werken met een declaratieve taal bij dit afstudeerproject:

- Een declaratieve taal is minder bekend. De misvatting bestaat dat het slechts een variant is van de bekende procedurele talen. Om met de declaratieve programmeer-

taal te kunnen werken is een geheel nieuwe kijk en studie op het programmeren nodig.

- Declaratieve talen zijn vaak niet erg goed voorbereid op numeriek rekenwerk: het implementeren hiervan is moeilijk. Mochten aan het huidige rekenprogramma rekenfuncties gekoppeld worden, dan zou dit wat moeilijker kunnen gaan dan bij een procedurele taal.

6.4 Verwijzing

Voor de in het voorgaande hoofdstuk besproken procedures is een programma geschreven in Prolog-2 voor Windows 3.1. Het programma zelf, commentaar bij de programmacode en een gebruiksaanwijzing bij het programma zijn te vinden in de bijlage "Gebruiksaanwijzing, code en toelichting bij het programma".

7 Conclusie

7.1 Doelstellingen

Als eerste is bij de hoofddoelstelling vermeld dat onderzoek gedaan zou worden naar de mogelijkheden om de computer als hulpmiddel te gebruiken bij het konstruktief ontwerpen, niet als rekentuig maar als werkelijke hulp bij het ontwerpproces. Zie voor een verdere uitleg 1.2.

Verder is in de nevendoelstelling de vraag gesteld of het mogelijk is om de techniek space-allocation -een methode om ruimtelijk te ontwerpen- te combineren is met de gevonden mogelijkheden om het konstruktief ontwerpen te ondersteunen.

7.2 Conclusie over doelstellingen

Als antwoord hierop kan vermeld worden dat er waarschijnlijk mogelijkheden zijn om de computer als hulpmiddel te gebruiken bij het konstruktief ontwerpen. Bij dit afstudeerproject is als deelresultaat een programma verkregen waarbij het duidelijk is dat een computer meer kan dan het uitvoeren van berekeningen. Wordt de nodige voorzichtigheid in acht genomen, dan is echter te konstateren dat het programma nog lang geen volwaardige hulp is bij het ontwerpproces. Of dit programma in de toekomst wel een volwaardige hulp voor het konstruktief ontwerpen zal zijn is niet met zekerheid vast te stellen, maar wel zeker is dat veel onderzoek in deze richting tot vruchtbare resultaten zal leiden.

Op de vraag over de mogelijke combinatie tussen space-allocation en de gevonden mogelijkheden om het konstruktief ontwerpproces te ondersteunen, kan eveneens positief geantwoord worden. De technieken kunnen gecombineerd worden en sluiten meerdere malen effectief elkaars te overvloedige mogelijkheden uit.

Merkwaardig is dat er geen conclusie getrokken kan worden over waar de grens ligt tussen ontwerpprocessen die wel vertaald kunnen worden naar een programma en die waarbij dat niet mogelijk is.

7.3 Ervaringen bij literatuurstudie/cases

- Bij de literatuurstudie komen de volgende zaken over het konstruktief ontwerpproces aan het licht:

  Ten eerste is het proces van belang zoals dat te zien is in figuur 15. Dit proces is aanwezig in alle cases en een vereenvoudiging van het proces dat in de literatuur beschreven is, te zien in de tweede afbeelding van figuur 1. Dit proces bevat impliciet veel informatie en handelingen.

In de literatuur is te lezen dat een van de problemen bij het omzetten van een konstruktief ontwerpproces in een algoritme is, dat met de scheppende rol van de mens in het ontwerpproces geen rekening gehouden wordt.

Het blijkt dat gekonstrueerd wordt met bepaalde klassen en verzamelingen konstruktie-elementen, te zien in de tweede afbeelding van figuur 2 en de eerste afbeelding van figuur 3.

Het blijkt dat de konstruktie-elementen veelvuldig in bepaalde geordende groepen voorkomen om een ruimte te overspannen, zie bijvoorbeeld de tweede afbeelding van figuur 4.

Het ruimtelijk ontwerp is een moeilijk vast te leggen proces en het resultaat wordt de konstrukteur meestal aangeboden.

In de cases is echter duidelijk dat een ruimtelijke vorm kan veranderen als de konstruktiekeuze daartoe aanleiding geeft. Een ruimte wordt bijvoorbeeld verplaatst. De ruimtelijk vorm geeft ook een voorwaarde voor de konstruktiekeuze.

Uit de literatuur is bekend dat niet hulpmiddelen ter systematisatie gebruikt moeten worden die reeds succesvol zijn bij de analyse van ontwerpen (bijvoorbeeld rekenapparatuur). Zie voor verder informatie over dit punt 3.3.

De mens gebruikt combinaties van kwalitatieve eisen bij het ontwerpen.

- Bij het beantwoorden van de vragen uit Bauwerk, Tragwerk, Tragstruktur, zie 3.4, komen enkele zaken over het konstruktief ontwerpproces aan het licht:

  Zoals geschematiseerd wordt, wordt ook vaak gekonstrueerd. Het wil echter niet zeggen dat zoals geschematiseerd wordt, ook gekonstrueerd moet worden.

  Er worden aannamen gedaan over het gedrag van konstrukties die tot gevolg hebben dat de ogenschijnlijke schematisatie van de konstruktie niet correct hoeft te zijn.

  In een konstruktief ontwerp-proces wordt of een poging gedaan de gunstigste draagsituatie van een konstruktie te verkrijgen oftwel aan esthetische eisen de voorkeur gegeven (of beide).

7.4 Ervaringen bij omzetting kennis in processen en data

- Een informatiemodel, in dit geval het Bouw Informatie Model, kan een goede hulp zijn bij het omzetten van kennis in processen en data, hierbij helpt vooral het

datamodel, maar meer in de zin van een goede representatie van de processen en data. Het feitelijke omzetten van de kennis in processen en data blijft, op de uitgangspunten en randvoorwaarden na, een creatief proces. Tegelijkertijd is het model niet geheel geschikt voor declaratieve doeleinden. Slechts het procedurele gedeelte van de gevonden algoritmen is goed te vertalen in een procesmodel.

7.5 Ervaringen bij omzetting processen en data in algoritmen

- Op dit punt aangekomen is duidelijk dat een gedeelte van het ontwerpproces is gestructureerd en in een algoritme/declaratief-regel-systeem is te vertalen.

- De techniek space-allocation mag zich na dit afstudeerproject verheugen op een nieuwe deeltechniek voor konstruktieve doeleinden:

  - het idee van de zonering
  - de konstruktief-voorwaarde-gebonden plaatsing van ruimten

- Hybride programmeren blijkt handig bij het programmeren, tot compacte programma's te leiden en goed toepasbaar te zijn bij deze groep van problemen. Helaas is de programmastructuur niet zo duidelijk vast te leggen als bij procedurele en in mindere mate puur declaratieve programma's.

7.6 Algemeen

Tot slot een samenvatting van de conclusies:

- Er is een antwoord op de vraag in de doelstellingen: bepaalde ontwerpdeelprocessen kunnen waarschijnlijk vervangen worden door programma's, en de techniek space-allocation is hiermee te combineren.

- Er is kennis verkregen over het konstruktief ontwerpproces

- Als gevolg hiervan is een programma geschreven met enkele interessante eigenschappen:

  Hybride (procedureel/declaratief) geprogrammeerd
  Nieuwe technieken als zonering en ruimteplaatsing met konstruktieve voorwaarden
  Konstrueert globaal ingevoerde of zelf gegenereerde ontwerpen i.p.v. rekenwerk
  De vorm beinvloed de konstruktie en de konstruktie beinvloed de vorm

In Bauwerk Tragwerk Tragstruktur [2] is te lezen:

"Het is te verwachten, dat in de nabije toekomst het gebruik en de beheersing van het beeldend vermogen in het ontwerpproces een nieuwe dimensie krijgt en veel vooruitgang, die nu nog als scheppende uiting van de menselijke intuitie gezien wordt, door middel van een algoritme aan de machine wordt overgelaten. De mens zal echter altijd de bron achter het ontwerpproces blijven."

Misschien dat dit afstudeerproject een kleine zet in de genoemde richting is.

Literatuurlijst

[1]
James E. Ambrose
Building Structures
ISBN 0-471-83094-1

[2]
Büttner Hampe
Bauwerk Tragwerk Tragstruktur, band 1 en 2
VEB Verlag für Bauwesen, Berlin

[3]
BIM Bouw Informatie Model

[4]
Wouter Ariens
Automatisering van het konstruktief-ontwerp proces
Afstudeerverslag T.U. Eindhoven

[5]
Walker, McCord, Sowa, Wilson
Knowlegde Systems and Prolog
ISBN 0-201-52424-4

[6]
Andrew Orton
The way we build now, form, scale and technique
ISBN 0 7476 0011 2

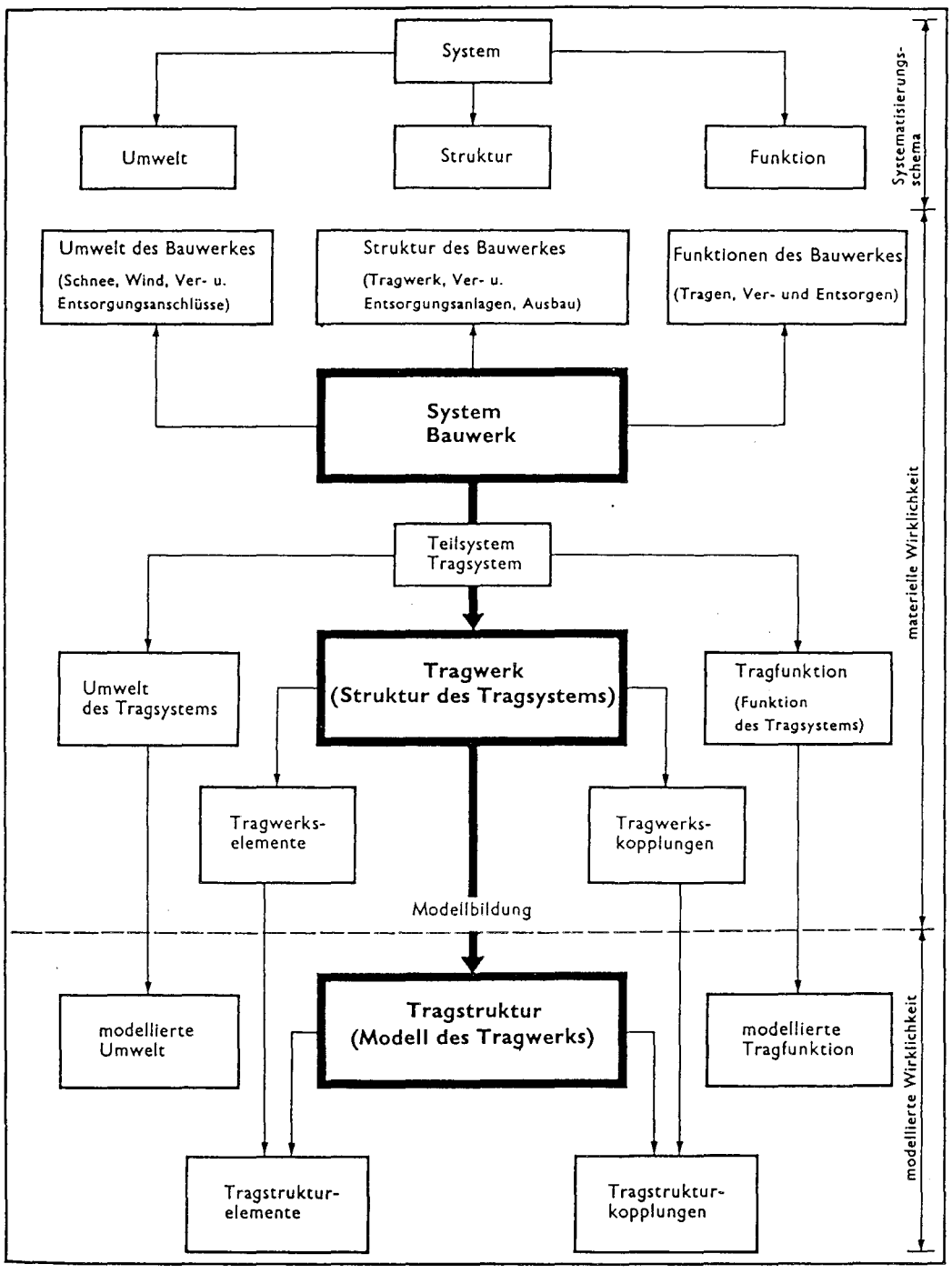
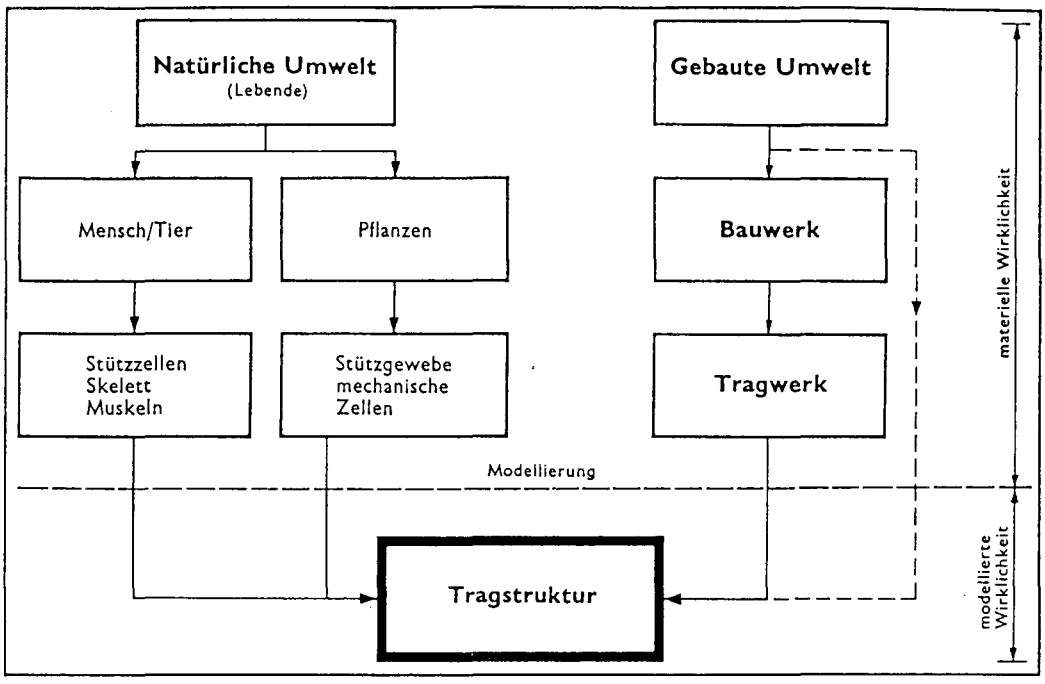


Fig. 1, eerste afbeelding, ordening van bouwwerk, draagwerk en draagsysteem

Fig. 1, tweede afbeelding, ontwerp van draagstrukturen



Tafel 1.1.3. Zuordnung von Tragfunktion, Tragqualität und Tragverhalten

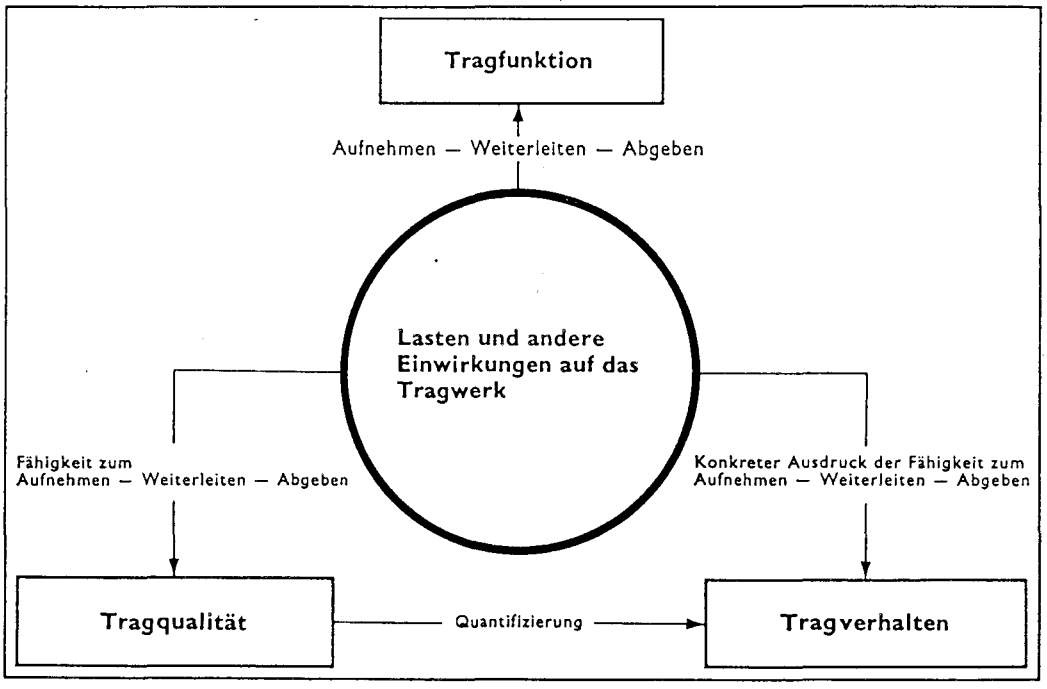
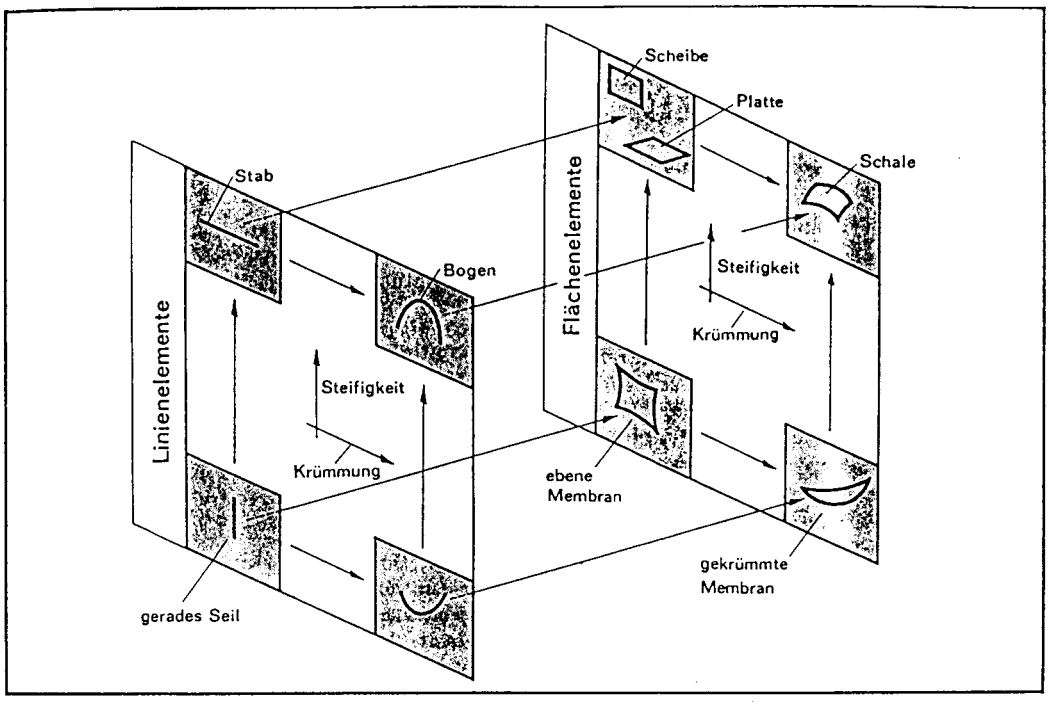


Fig. 2, eerste afbeelding, elementen van draagwerken



Tafel 1.2.3. Klassifizierung der Tragstrukturen

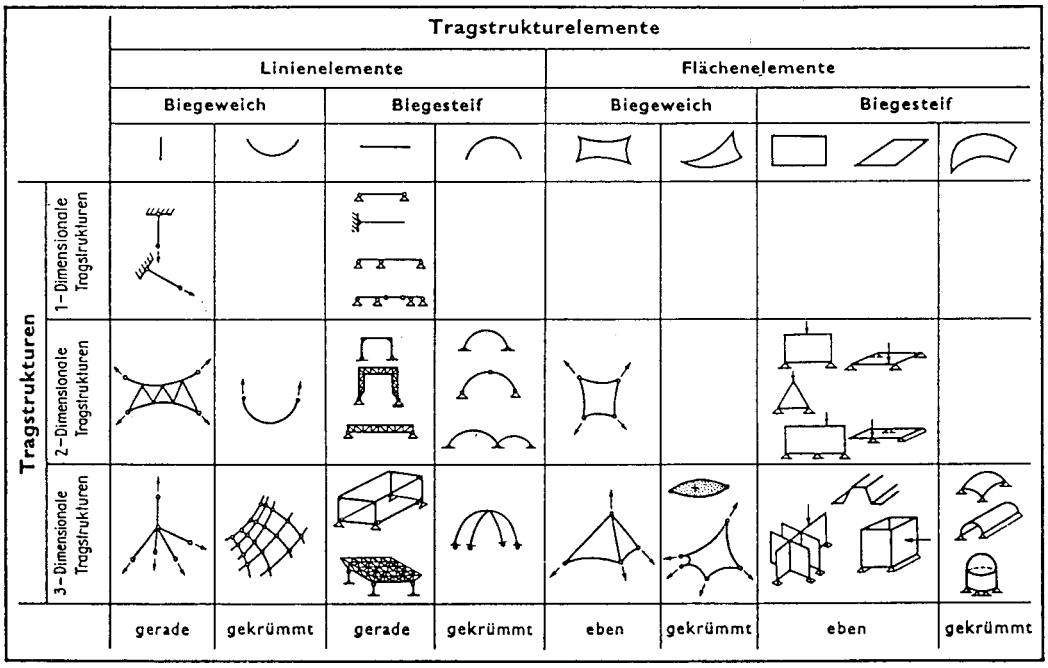


Fig. 2, tweede afbeelding, klassificering van elementen

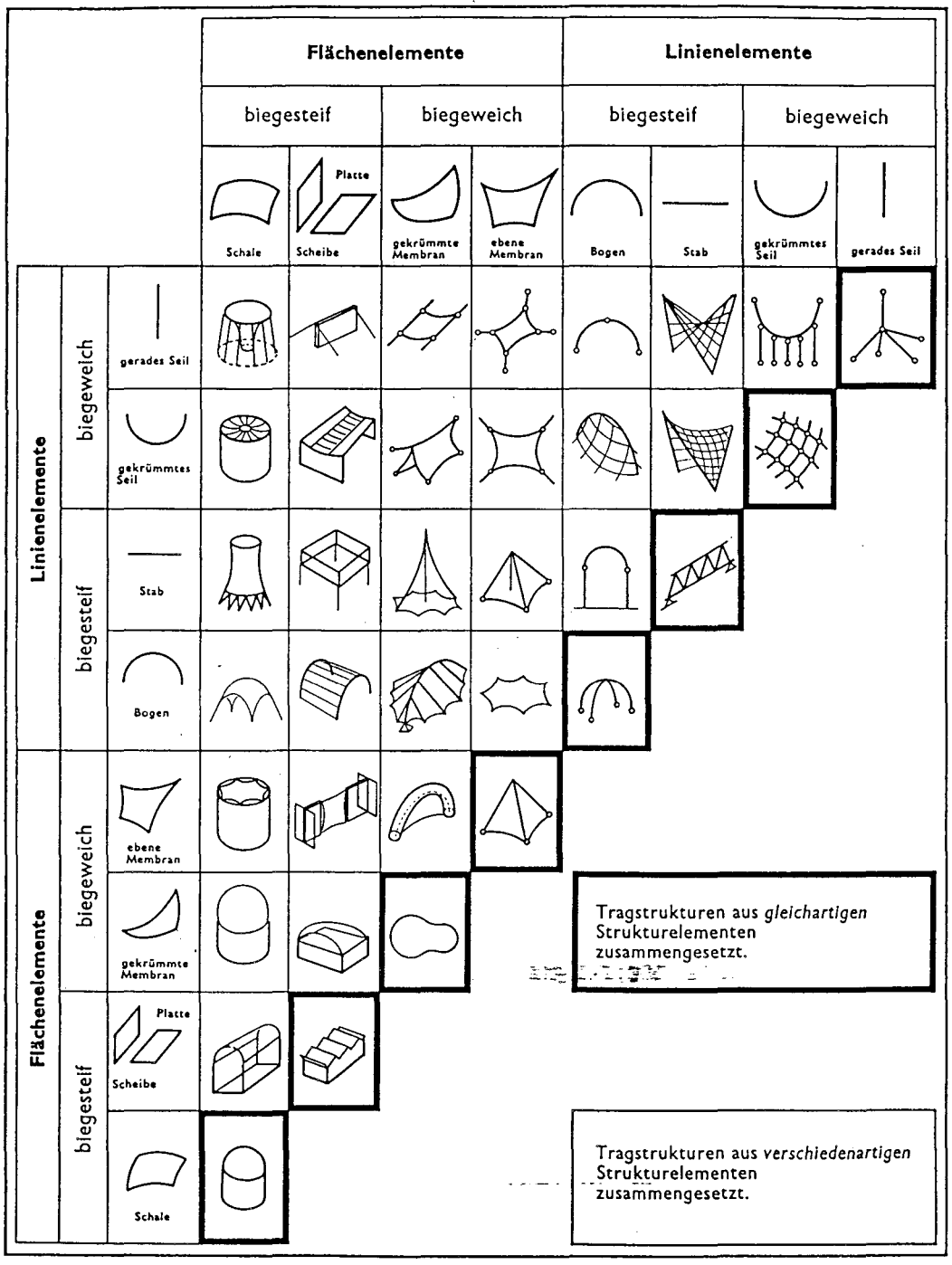


Fig. 3, eerste afbeelding, combinatie van struktuurelementen

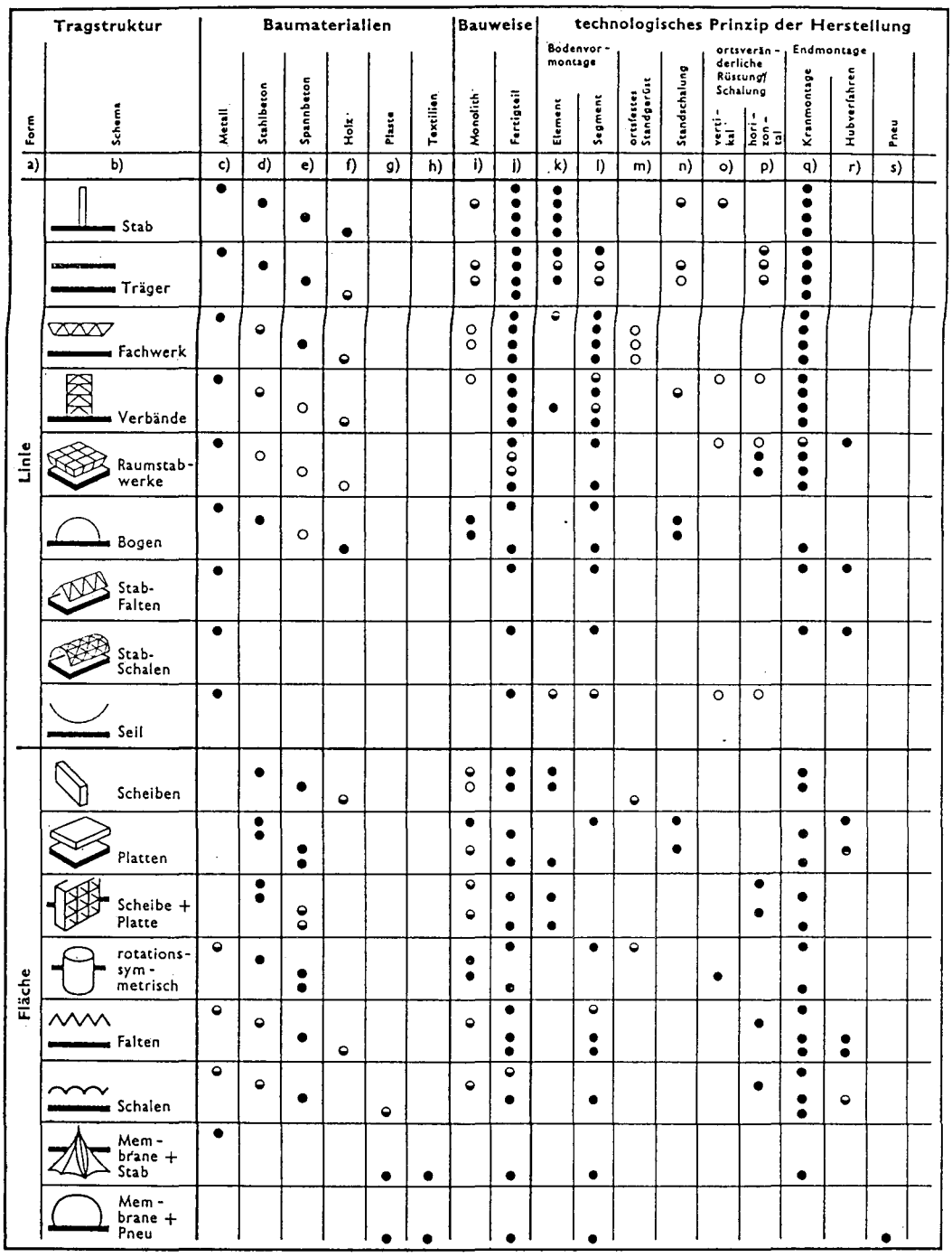


Fig. 3, tweede afbeelding, draagstrukturen

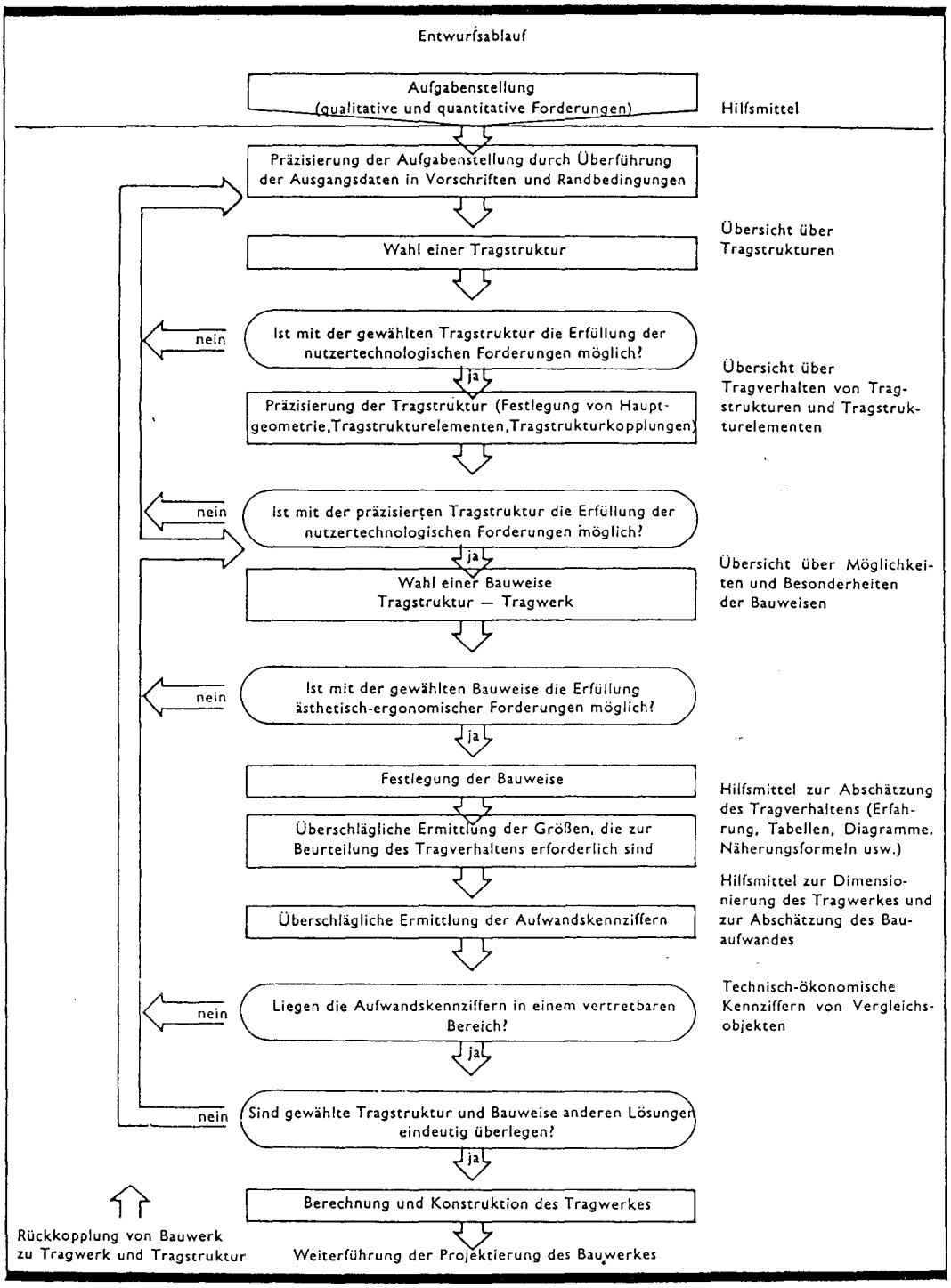


Fig. 4, eerste afbeelding, ontwerp van draagstrukturen 2

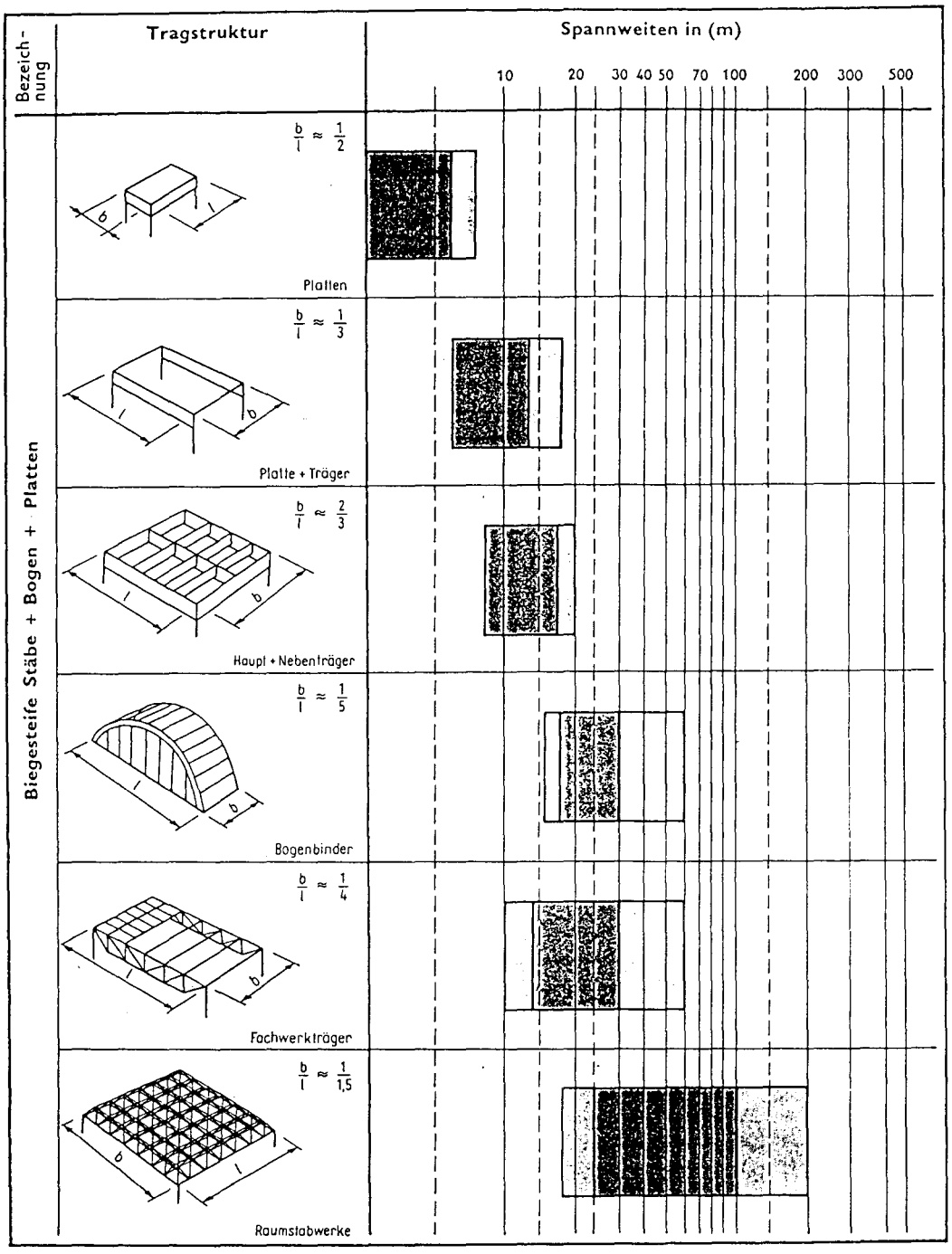


Fig. 4, tweede afbeelding, toepassingsmogelijkheden van bepaalde draagstrukturen

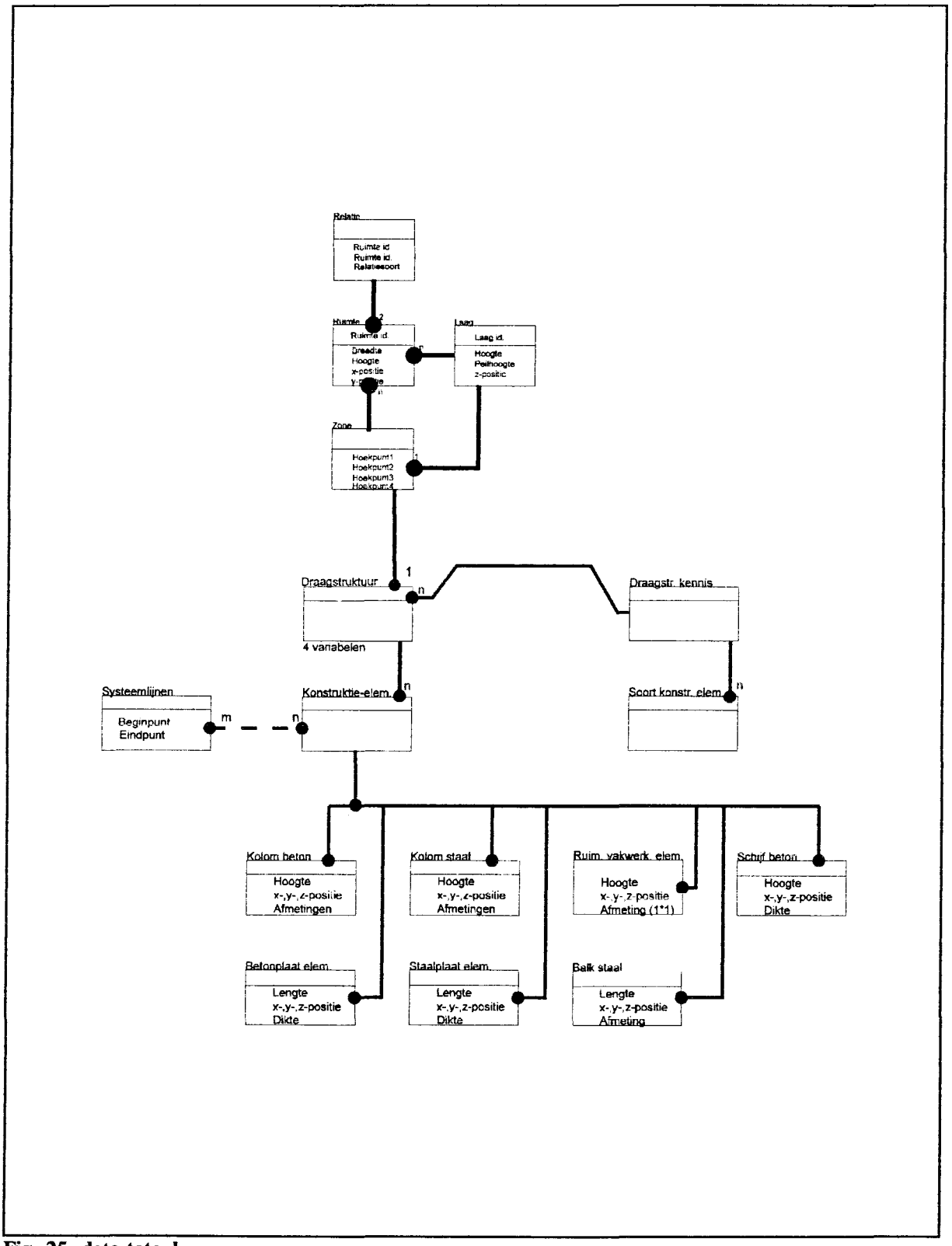


Fig. 25, data totaal

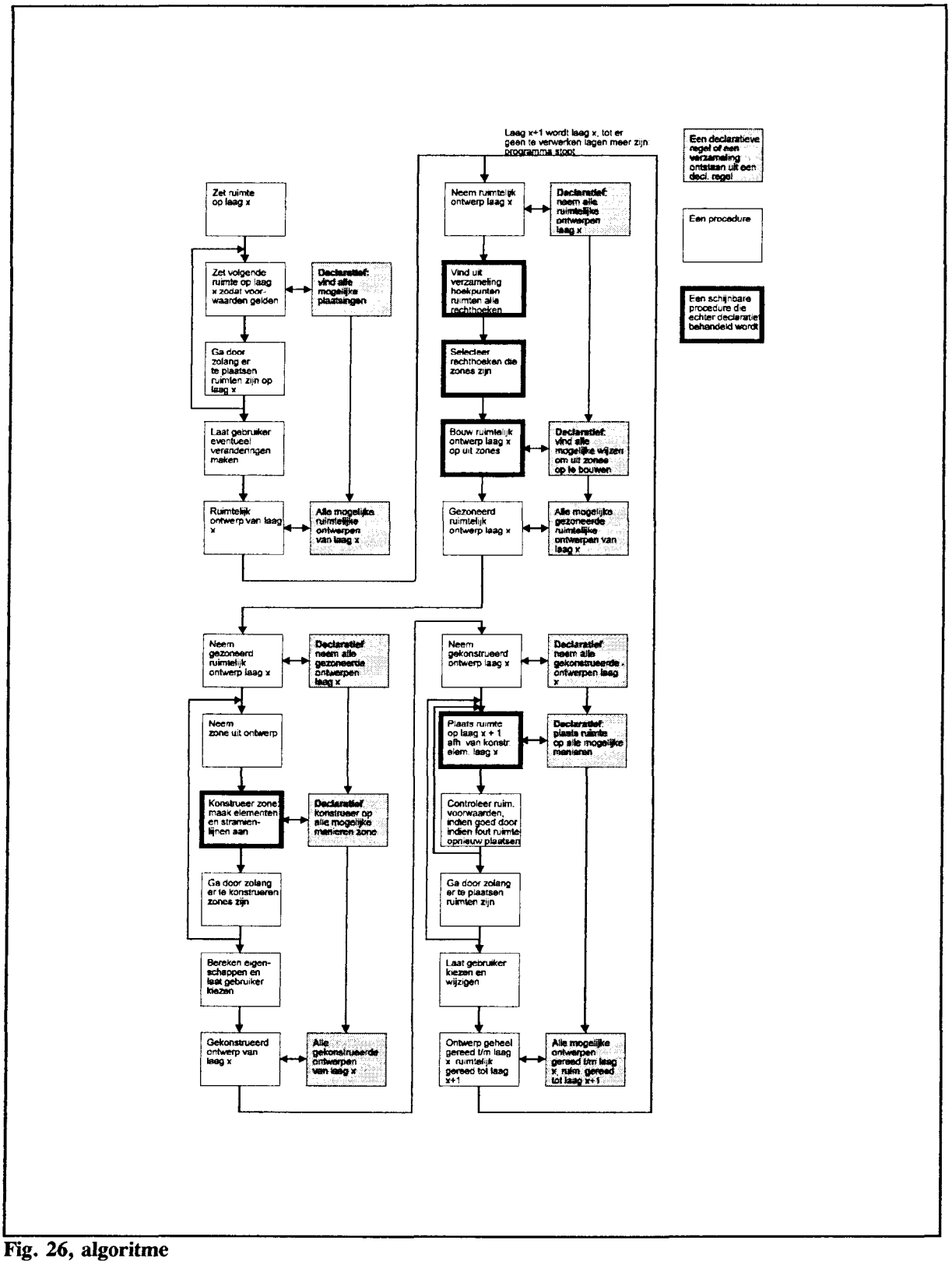


Fig. 26, algoritme

Faculteit Bouwkunde
Vakgroep
Konstruktief
Ontwerpen

Technische Universiteit
Eindhoven

Konstruktief ontwerpen met behulp van computerprogrammatuur

Bijlage Gebruiksaanwijzing, Code en toelichting bij programma

Begeleidingscommissie:

Prof.dr.ir. H.S. Rutten
Prof.ir. A.J.M. Beulens
Ir. P.H. van Merendonk
Ir. H.J. Fijneman

Errata

bij Hofmeyer, H.: Konstruktief ontwerpen met behulp van computerprogrammatuur, (2) Bijlage Gebruiksaanwijzing, code en toelichting bij programma, Technische Universiteit Eindhoven, Department of Architecture, Building and Planning, Structural Design Group, 1994.

Blz. 12

Bij het predicaat "invoer_ruimten:-", bij de laatste regel een uitroepteken toevoegen. De laatste regel wordt dan:

```
(E=vw,invoer_vw)),!.
```

Bij het predicaat "invoer_vw:-", bij de laatste regel een uitroepteken toevoegen. De laatste regel wordt dan:

```
(D=b,invoer_laag)),!.
```

Blz. 13

Bij het predicaat "invoer_laag:-" (nog op blz. 12), bij de laatste regel (op blz. 13) een uitroepteken toevoegen. De laatste regel wordt dan:

```
(D=k)),!.
```

Blz. 16

De laatste regel:

```
vorm_goed_lijsten(X,[]).
```

dient voor (en dus niet achter) de voorgaande drie regels te staan, dus:

```
vorm_goed_lijsten(X,[]).
vorm_goed_lijsten(X,Lijst):-pak2(T,Lijst,Nlijst),
maak_goed_lijst(T,X,Lijst2),assert(specialelijst(Lijst2)),
vorm_goed_lijsten(X,Nlijst).
```

Blz. 17

Eerste drie programmaregels dienen vervangen te worden door:

```
prima(T,X):-pos(T,[_,_,_]),voorwaarde_univers(_,T,X).
```

Blz. 29

Voor het predicaat "z_verwerk" dient een regel te worden toegevoegd met het volgende resultaat:

```
z_verwerk([],NNlijst,NNlijst).

z_verwerk(Lijstzones,Nlijst,Sortlijst):-
enz.
```

Eindhoven, 6 juni 2005, Herm Hofmeyer

Inhoudsopgave

Inleiding en doelstellingen

Inleiding

Deze bijlage dient in de eerste plaats als een gebruiksaanwijzing bij het computerprogramma "Blok", behorende bij het afstudeerproject "Konstruktief ontwerpen met behulp van computerprogrammatuur".

De programmacode, geschreven in Prolog-2 voor Windows 3.1 (afgekort: P2W3), wordt in deze bijlage behandeld en toegelicht.

P2W3 is een declaratieve programmeertaal die functioneert met het besturingssysteem Windows 3.1. Er wordt vanuit gegaan dat de gebruiker van het programma Blok met dit besturingssysteem overweg kan. Verder is er enige vaardigheid nodig in het omgaan met de programmeeromgeving, dat wil zeggen dat de gebruiker een programma, geschreven in Prolog, moet kunnen opvragen en kunnen opstarten. Nietemin zal dit laatste kort behandeld worden in deze gebruiksaanwijzing.

Gebruiksaanwijzing

Algemeen

Het programma is geschreven in:

Prolog-2 voor Windows 3.1 (afgekort: P2W3)

De leverancier van deze programmeeromgeving is:

Expert Systems International BV
Kersbergenlaan 4a
Postbus 148
3700 AC ZEIST
03404-14460

Installatie

Windows 3.1. dient geinstalleerd te zijn volgens de daartoe bestemde procedures gegegeven bij dit besturingssysteem. Een resolutie van 1024 * 768 pixels wordt ten sterktste aangeraden omdat anders diverse grafische afbeeldingen in het programma "Blok" slechts ten dele worden weergegeven. De resolutie is afhankelijk van de grafische kaart in de computer en afhankelijk van de gebruikte "driver" voor deze kaart.

P2W3 dient onder Windows 3.1. geinstalleerd te worden volgens de instructies gegeven door Expert Systems International. Deze instructies zijn aanwezig bij het programma, dat beschikbaar is bij de beheerders van het computernetwerk van BKO.

Een korte samenvatting van de installatie van P2W3:

1) Installeer P2W3 door de file "INSTALL.EXE" van de installatieschijf onder File-Manager te runnen.

2) Kies "Install" onder het menu zoals dat verteld wordt bij de installatie en volg eventuele instructies op.

3) Ga uit Windows en run onder D.O.S. de file "UPDATE.EXE" van de installatie-schijf.

4) P2W3 is nu geinstalleerd.

Het programma "Blok" bestaat uit 3 kleine files en deze files zijn beschikbaar op de diskette bij deze bijlage:

PRO1.PRO (15451 Bytes)
TEK1.PRO (29481 Bytes)
KON1.PRO (14901 Bytes)

Tezamen 58,4 kBytes.

Het programma bestaat uit 3 files omdat er beperkte mogelijkheden zijn om files te veranderen of te maken binnen P2W3: deze te veranderen of te maken files mogen niet te groot zijn. Het is mogelijk om het programma in 1 file te veranderen, maar praktisch is dit niet nodig. Bij de installatie van P2W3 worden de files van P2W3 in een directory "PROLOG2" geplaatst. Het is handig als de drie kleine files van "Blok" ook in deze directory staan. Daartoe kan in D.O.S. de volgende opdracht gegeven worden, al is deze procedure ook uit te voeren met de File-manager van Windows:

COPY A:*.PRO C:\PROLOG2

Starten

Na P2W3 opgestart te hebben onder Windows door op het desbetreffende icoon twee maal te klikken, is een window "Prolog2" te zien met diverse hoofdmenu's. Met hoofdmenu's worden hier de menu's bedoeld die direct zichtbaar zijn bij het programma, zoals de menu's File, Edit, Search, Window, enz.

Zoals bekend wordt verondersteld kan in het besturingssysteem Windows een window gemaximaliseerd en geminimaliseerd worden in grootte, en bovendien verplaatst en in willekeurige grootten vervormd worden. Een actieve window is bij de standaard kleureninstellingen herkenbaar aan een blauwe kop terwijl de andere windows een witte kop hebben. Een window is actief te maken door op de titel te klikken.

Binnen de window "Prolog2" zullen allerhande windows te zien zijn waarbij het soms handig is om alle windows te zien. Dit kan door onder het hoofdmenu Window van "Prolog2" te kiezen voor "Tile".

Windows waarin de programmatekst veranderd kan worden en zichtbaar is noemen we edit-windows.

Bij elk van de 3 programmadelen van "Blok" is een edit-window te maken door te kiezen bij het hoofdmenu File voor "Open". Nu is te kiezen uit een aantal bestanden waarbij, indien de juiste directory "Prolog2" gekozen is, ook de bestanden "PRO1.PRO", "TEK1.PRO" en "KON1.PRO" zichtbaar zijn. Door een programma te kiezen wordt hiervoor een edit-window gemaakt: het programma is nu zichtbaar.

Indien voor alle drie programma's een edit-window bestaat, kunnen we deze programma's in de P2W3-database invoeren door een edit-window actief te maken (door te klikken op de titel, vervolgens zal de titel blauw kleuren bij de standaard kleureninstellingen) en vervolgens bij het hoofdmenu Program voor de optie "Consult" te kiezen. Waarom deze stappen genomen worden is verweven met kennis omtrent declaratieve talen en hun programma-opzet en wordt hier niet besproken. Alle drie de programma's dienen zo in de database opgenomen te worden. Door voor de reeds besproken optie Tail uit het hoofdmenu Window te kiezen blijft het overzicht tussen de verschillende windows gewaarborgd.

Een andere mogelijkheid om de drie programma's in de database van P2W3 in te voeren is de window "User" actief te maken. Hierna worden de volgende regels getypt, allen gevolgd door een return:

consult(tek1). <return>
consult(pro1). <return>
consult(kon1). <return>

Nu kan opnieuw naar de window "User" gegaan worden (deze is reeds actief na de diverse consult-aanroepen). Achter het vraagteken typt men:

run. <return>

Het programma "Blok" is nu actief.

Werken met "Blok"

Tijdens het werken met het programma "Blok" zijn enkele zaken van belang:

- Elke invoer bij het programma moet volgens de voorwaarden die bij de invoer gegeven worden geschieden, gevolgd door een punt en hierna gevolgd door een return. De invoer van bevoorbeeld een lengte van 6 meter ziet er dan als volgt uit: <6><.><return>, waarbij de haken de toetsaanslagen aangeven.

- Het is mogelijk dat een "Could not draw graphic"-error gemeld wordt. Dit is geen fout van het programma "Blok" maar een P2W3-codefout. Opnieuw beginnen is de enige remedie.

- Geregeld kan een melding gemaakt worden van een "Prolog2-error" omdat een integer getal wordt afgerond door de functie "fix". Klikken op de knop "Ok" is voldoende om het programma verder goed te laten functioneren.

- De gebruiker zal zelf de formaten van de windows moeten aanpassen en windows op de voor- of achtergrond moeten zetten om duidelijkheid te bevorderen. P2W3 zorgt er wel voor dat een actieve window altijd te zien is. De genoemde optie "Tile" onder het hoofdmenu "Window" kan helpen een goed overzicht van de gebruikte windows te krijgen.

- Is het wenselijk tijdens het programma te stoppen, dan kan overal waar iets ingevoerd moet worden het volgende worden gedaan. Houd de control-toets ingedrukt en toets de letter "b". Toets hierna een punt en return. Samengevat: <CTRL><b><.><return>. Er verschijnt een klein menu waar de radiobutton "Abort" moet worden ingedrukt. Druk hierna "Ok". Eventueel deze procedure nogmaals herhalen indien weer een invoer gevraagd is, herkenbaar aan een knipperende cursor in de window "User". Hierna kan met de optie "Exit" bij het hoofdmenu "File" teruggekeerd worden naar Windows.

Invoer

Nadat het programma echt gestart is, is de volgende tekst leesbaar op het scherm (tekst op het scherm wordt hier met een ander lettertype aangegeven):

Programma Blok, versie 1

Invoer ruimten en voorwaarden

Na de gewenste invoer een . en een return invoeren, bijvoorbeeld:
keuken. <return>
3. <return>

enz.

Voor afmetingen alleen de even getallen gebruiken.

Gebruik voor lagen nummers, beginnende met 1, geef laag 1 peilhoogte 0.

Er komen ook konstruktievarianten voor met zones niet gekonstrueerd, dus ook varianten in het geheel niet gekonstrueerd.

Voer ruimtenaam in: _

Nu kan een ruimtenaam ingevoerd worden, aangegeven door het streepje, gevolgd door een punt en een return, waarna het systeem zal vragen naar de afmetingen van de ruimte en de laag waarop deze ruimte ligt. Het woord "waarop" moet niet te letterlijk genomen worden. Er wordt gevraagd in welke laag de ruimte ligt, van welke laag de ruimte deel uit maakt.

Breedte van x : _
Lengte van x : _
Laag waarop x ligt: _

Nu is de mogelijkheid daar om nog een ruimte in te voeren of om voorwaarden in te voeren. Het is niet mogelijk later nog een ruimte in te voeren. Alle ruimten, op alle lagen dienen te worden ingevoerd voordat het invoeren van de voorwaarden kan beginnen.

Volgende ruimte invoeren: typ r.
Voorwaarden invoeren: typ vw.
_

Wordt voor de eerste mogelijkheid gekozen, dan volgt nogmaals de invoer van de gegevens van een ruimte. Wordt voor de tweede mogelijkheid gekozen, dan kunnen voorwaarden ingevoerd worden, zoals hieronder te zien is. In versie 1 van het programma "Blok" is het alleen mogelijk voorwaarden te noemen tussen twee ruimten waarbij de voorwaarde bestaat uit het feit dat de twee ruimten naast elkaar moeten liggen, dit is de voorwaarde "a", een afkorting van de naam "aanliggend".

Voer ruimtenaam 1 in: _
Voer ruimtenaam 2 in: _
Voer voorwaarde in (typ bij versie 1 alleen a): _

Na deze invoer geeft het programma de keuze om nog meer voorwaarden in te voeren of om informatie over de bouwlagen in te voeren, door de onderstaande uitvoer.

Volgende voorwaarde invoeren: typ vw.
Informatie bouwlaag invoeren: typ b.
_

Wordt voor de eerste mogelijkheid gekozen, dan volgt nogmaals de invoer van een voorwaarde. Wordt voor de tweede mogelijkheid gekozen dan is de invoer mogelijk van informatie over de bouwlagen. Een bouwlaag is synoniem aan een verdieping. Het is noodzakelijk de bouwlagen opeenvolgend te nummeren van 0 tot de hoogste laag, waarbij de onderste laag de laag wordt met 0 en de bovenste laag de laag met het hoogste getal. Verder dient de hoogte van een laag opgegeven te worden waarmee de verdiepingshoogte bedoeld wordt en dient opgegeven te worden hoe hoog de vloer van een laag ligt ten opzichte van de grond, dit wordt in het programma de peilhoogte genoemd. De invoer van de informatie over de lagen mag wel in willekeurige volgorde van de lagen geschieden. Ook hier dienen, aangezien het afmetingen betreft, gehele getallen gebruikt te worden. Even getallen, zoals bij de afmetingen van ruimten, hoeven niet verplicht gebruikt te worden: ook oneven getallen zijn toegestaan.

Voer bouwlaagnummer in: _
Voer bouwhoogte van laag x in: _
Voer peilhoogte (niveau) van vloer bouwlaag x in: _

Na deze invoer kan gekozen worden om de informatie van een andere laag in te voeren. Zijn alle lagen reeds ingevoerd dan kan verder gegaan worden met het programma.

Volgende laag invoeren: typ l.
Klaar met invoer: typ k.

_

Interactie

Na het gereed komen van de invoer is op het scherm, in de window "User", de volgende mededeling te zien met onder deze mededeling een aantal sterretjes, die aangegeven hoeveel ruimtelijke ontwerpen geproduceerd zijn:

Ruimtelijk ontwerp eerste laag

Nadat het eerste sterretje verschijnt, is een ruimtelijk ontwerp van de eerste laag gegenereerd. Er wordt een window "plattegrond" gemaakt waarin de plattegrond van het ontwerp te zien zijn. De ruimten worden aangegeven met groene lijnen. De naam van de ruimte staat in de ruimte met zwarte tekst. De gebruiker de kans krijgt interactief te reageren aan de hand van de volgende boodschappen op het scherm:

Ontwerp niet meenemen naar volgende ontwerpstap: a.
Ontwerp meenemen naar volgende ontwerpstap: b.
Ontwerp aanpassen en daarna meenemen naar volgende ontwerpstap: c.
Alle nu nog komende ontwerpen inclusief deze overslaan en door naar volgende ontwerpstap:

Uw keuze: _

Bij het kiezen van optie a zal een volgend ruimtelijk ontwerp van de eerste laag gemaakt worden, zonder dat het zojuist gepresenteerde ontwerp straks als een basis kan dienen voor het zoneringsproces: het ontwerp wordt eenvoudig "vergeten" als mogelijkheid.

Bij het kiezen van optie b zal een volgend ruimtelijk ontwerp van de eerste laag gemaakt worden, maar hier zal het zojuist gepresenteerde ontwerp opgeslagen worden en straks bij het zoneringsproces als een mogelijkheid gebruikt worden.

Bij het kiezen van optie d stopt het ruimtelijk ontwerpproces van de eerste laag en wordt doorgegaan naar het zoneringsproces.

Bij het kiezen van optie c kan het zichtbare ontwerp in window "plattegrond" gewijzigd worden door 1 van de ruimten een andere plaats te geven. De plaats die door de gebruiker gegeven kan worden is geheel vrij, dat wil zeggen dat met de voorwaarden bij de invoer nu geen rekening gehouden wordt, indien de gebruiker dit zelf niet doet. Het is dus mogelijk om ruimten te laten overlappen of los van elkaar te laten liggen. Bij het kiezen van optie c verschijnen de volgende mededelingen, waarbij tussen haakjes informatie verschijnt over de ruimtenamen en de bij de ruimten behorende posities:

Het wijzigen van een ontwerp kan op grond van flexibiliteit zonder rekening te houden met de voorwaarden bij de invoer. Let goed op dat ruimten elkaar bijvoorbeeld kunnen overlappen of los van elkaar kunnen liggen.

[...]
[...]

Welke ruimte wilt U wijzigen van plaats, geef alleen de naam van de ruimte gevolgd door een punt en een return: _

Na het ingeven van de ruimtenaam, wederom gevolgd door een punt en een return, vraagt het programma naar de nieuwe positie van de ruimte door de volgende vragen:

Geef de x-waarde van de nieuwe positie van ruimte x: _
Geef de y-waarde van de nieuwe positie: _

Na de invoer van de nieuwe posities van de te verplaatsen ruimte wordt in de window "plattegrond" opnieuw de plattegrond getekend. Als de gebruiker deze plattegrond gezien heeft kan naar het volgende ruimtelijk ontwerp gekeken worden door in te gaan op de volgende mededeling:

Gezien, toets willekeurig teken, punt en return: _

Nu kan een willekeurig teken ingevoerd worden om te zorgen dat naar het volgend ruimtelijk ontwerp gekeken kan worden. Zolang er ruimtelijke ontwerpen te genereren zijn, zal het programma telkens een ruimtelijk ontwerp laten zien en de gebruiker laten kiezen voor

de genoemde opties a,b,c en d. Zijn er geen mogelijkheden meer om een ruimtelijk ontwerp voor de eerste laag te laten kiezen dan wordt automatisch overgestapt naar het zoneren van de ruimtelijke ontwerpen. Op het scherm staat nu:

Zoneren van laag x

Is het ruimtelijk ontwerpen van laag 0 gereed dan zal laag x gelijk zijn aan laag 0. Is bijvoorbeeld een hele laag 2 afgehandeld, dat wil zeggen ruimtelijk ontworpen, gezoneerd en gekonstrueerd, en is het ruimtelijk ontwerp van laag 3 ook afgehandeld, dan zal laag x gelijk zijn aan laag 3. Ook onder deze regel in de window "User" verschijnen sterretjes die vertellen hoeveel gezoneerde ontwerpen reeds geproduceerd zijn.

Nadat alle ruimtelijke ontwerpen gezoneerd zijn, waarbij het goed mogelijk is dat er meer gezoneerde ontwerpen ontstaan dan er ruimtelijke ontwerpen zijn, wordt automatisch overgeschakeld op het konstrueren van de betreffende laag. Dit is te zien aan de mededeling:

Konstrueren van laag x

Ook onder deze mededeling verschijnen sterretjes die aangegeven hoeveel gekonstrueerde ontwerpen gereed zijn. Als een konstruktief ontwerp van een laag gereed is, wordt een window "ruim_ontwerp" gemaakt, waarin het gehele ruimtelijke ontwerp tot de dan toe ontworpen lagen, drie-dimensionaal getekend. Hierna wordt een window "ko_ontwerp" gemaakt waarin het konstruktieve ontwerp getekend wordt. Hier worden wel konstruktie-elementen getekend, maar altijd met dezelfde afmetingen. Het varieren van afmetingen van konstruktie-elementen is dus niet zichtbaar in de window "ko_ontwerp". De elementen zijn door hun kleur en vorm snel te herkennen wat betreft materiaal en vorm. Vervolgens worden in de tekening van het ruimtelijk ontwerp, in de window "ruim_ontwerp", alle stramienlijnen getekend van de konstruktie met bijbehorend begin- en eindpunten van de stramienlijnen, weergegeven door de x-, y- en z-positie tussen haakjes. Nadat de algoritmen bepalen hoeveel kilo staal en beton in het konstruktief ontwerp zitten, komt de volgende mededeling op het scherm:

In dit ontwerp is circa x kg. staal aanwezig. In dit ontwerp is circa y kg. beton aanwezig.

Konstruktief ontwerp meenemen naar volgende ontwerpstap: a.
Konstruktief ontwerp niet meenemen: b.

Na het zien van de gegevens en het konstruktief ontwerp kan de gebruiker beslissen of het konstruktief ontwerp wordt meegenomen naar de volgende ontwerpstap, hetgeen betekent dat op het konstruktief ontwerp zoals dat nu te zien is in de window "ko_ontwerp" de ruimten worden geplaatst van de volgende te plaatsen laag indien, dat is vanzelfsprekend, er nog te plaatsen ruimten op een volgende laag zijn.

Bij het kiezen van optie a wordt het volgende konstruktief ontwerp gepresenteerd door middel van nieuwe tekeningen in de windows "ruim_ontwerp" en "kon_ontwerp" en wordt het huidige konstruktief ontwerp "onthouden" en gebruikt bij de volgende ontwerpstap.

Bij het kiezen van optie b wordt het volgende konstruktief ontwerp gepresenteerd door middel van nieuwe tekeningen in de windows "ruim_ontwerp" en "kon_ontwerp" en wordt het huidige konstruktief ontwerp "vergeten" en niet gebruikt bij de volgende ontwerpstap.

Zijn alle konstruktieve ontwerpen zo behandeld, dan kan doorgegaan worden met het ruimtelijk ontwerpen van de volgende laag. De mededeling op het scherm is:

Ruimtelijk ontwerpen van laag x

Onder deze mededeling verschijnen sterretjes om het aantal gerealiseerde ruimtelijke ontwerpen weer te geven. Bij een gereed ruimtelijk ontwerp zal een window "plattegrond" gemaakt worden waarin met rode lijnen het ontwerp van de huidige beschouwde laag te zien is. Met groene lijnen is de laag hieronder aangegeven, zodat enig inzicht verkregen wordt over de laagopbouw. Er verschijnt op het scherm:

De groene ruimten liggen op de laag onder de nu te ontwerpen laag,
de rode ruimten zijn nu geplaatst.

Ontwerp niet meenemen naar volgende ontwerpstap: a.
Ontwerp meenemen naar volgende ontwerpstap: b.
Ontwerp aanpassen (rood aangegeven ruimten) en daarna meenemen naar volgende ontwerpstap: c.
Alle nu nog komende ontwerpen inclusief deze overslaan en door naar volgende ontwerpstap: d.

Uw keuze:

Verder gaat het programma door zoals beschreven bij het gedeelte ruimtelijk ontwerp. Deze cyclus gaat een aantal malen door totdat alle lagen behandeld zijn. Hierna stopt het programma, alle ontwerpen zijn immers gepresenteerd.

Door het kiezen van de optie "Exit" bij het hoofdmenu "File" kan teruggekeerd worden naar Windows. Is het wenselijk dat het programma meerdere keren achter elkaar gebruikt wordt, dan zal eerst teruggekeerd moeten worden naar Windows, waarna opnieuw P2W3 en de drie programma's opgestart kunnen worden.

Code en toelichting

Inleiding

Het programma, bestaande uit de kleine programma's PRO1, KON1 en TEK1, is hieronder te zien in een ander lettertype. Het commentaar is in het gewone lettertype tussen de programmaregels verwerkt. De drie programma's zijn vrij willekeurige opdelingen van de totale code van het programma. Bij de consult-opdrachten in Prolog zullen alle programmaregels toch netjes samen in de database terecht komen. Hoewel het toevallig is dat een bepaalde programmaregel (predicaat) in een bepaald klein programma staat, zullen we beginnen met PRO1. PRO1 bevat de hoofdstructuur van het programma en enkele hulppredicaten.

PRO1

```
lijstomdraai([],Eindlijst,Eindlijst).

lijstomdraai(Lijst1,Tlijst,Eindlijst):-
pak2(Element,Lijst1,Lijst2),
voegbij(Element,Tlijst,NTlijst),
lijstomdraai(Lijst2,NTlijst,Eindlijst).

ruimte3(P,N):-
ruimte(P,_,_,N).

voegbij2(A,[],[A]).

voegbij2(A,Lijst,[A|Lijst]).

append([],L,L).

append([H|T],L,[H|T2]):-append(T,L,T2),!.

voegbij([A],[],[A]).
voegbij(A,Lijst,[A|Lijst]).

pak(A,[A|B],B).

pak(A,[B|C],[B|E]):-pak(A,C,E).

pak2(A,[A|B],B).

pak3(A,[A|B],B).

pak3(A,[B|C],D):-pak3(A,C,D).

pick(B,[B|Staart]).

pick(B,[A|Staart]):-pick(B,Staart).

ruimte2(A):-ruimte(A,_,_,1).
```

```
pos2(B):-pos(B,_).

count(1).

count(I):-count(J),I is J+1.

count(I,N):-count(I),(I>N,!,fail;true).

abs(A,B):-A>=0, B is A,!.

abs(A,B):-A<0, B is -A,!.
```

"lijstomdraai", "voegbij2", "voegbij", "pak", "pak2", "pak3", "pick", "ruimte3", "ruimte2", "pos2", "count" en "abs" zijn veel gebruikte en eenvoudige predicaten. Ze zijn alle behandeld in de bijlage "T8-T9-verslagen" en in het boek "Knowlegde systems and Prolog" van Walker. Het predicaat "lijstomdraai" wordt gebruikt om alle elementen in een lijst om te draaien: het achterste element van de lijst wordt het voorste element en het voorste element in een lijst wordt het achterste.

```
run:-
invoer,
rol1,
zon(1),
kon(1),
cycle(N).

cycle(N):-onderzoek(Q), QQ is Q-1, count(N,QQ),P is N+1, cycle2(P), fail.

cycle2(N):-ro(N),zon(N),kon(N).

cycle(N).
```

Het predicaat "run" wordt aangeroepen vanuit de interference-engine van de programmeeromgeving van Prolog. In deze omgeving kunnen predicaten worden ingetoetst waarna de interpreter deze predicaten probeert waar te maken. Om run waar te maken worden een aantal regels op het scherm getoond met "write", en "nl" dient om een regel over te slaan. Hierna wordt "invoer" aangeroepen. Als "invoer" slaagt kan de eerste laag ruimtelijke ontwerpen gegenereerd worden met "roll". Hierna wordt laag 1 gezoneerd met "zon" en gekonstrueerd met "kon". Hierna wordt "cycle" aangeroepen die cyclisch een laag zal ontwerpen, zoneren en konstrueren. Met het predicaat "onderzoek" wordt het hoogste lagennummer gezocht. Vervolgens wordt geteld vanaf 1 tot dit hoogste lagennummer met "count". "cycle2" zal dan deze laag dan ontwerpen met "ro", zoneren met "zon" en konstrueren met "kon". Deze laatste drie predicaten hebben 1 variabele: het lagennummer van de laag waar het predicaat betrekking op heeft.

```
invoer:-
nl, write("Programma Blok, versie 1"),
```

```
nl,nl,
nl, write("Invoer ruimten en voorwaarden"),
nl,nl, write("Na de gewenste invoer een .  en een return
invoeren, bijvoorbeeld:"),
nl, write("keuken. <return>"),
nl, write("3. <return>"),
nl, write("enz."),
nl, write("Voor afmetingen alleen de even getallen, voor konstruktieve var."),
nl, write("gebruik van 0 tot 4 (dus eigenlijk alleen 2 en 4)."),
nl,nl, write("Gebruik voor lagen nummers, beginnende met 1, geef laag 1 peilhoogte 0."),
nl,nl, write("Er komen ook konstruktievarianten voor met zones niet gekonstrueerd, dus ook varian-
ten"),
nl, write("in het geheel niet gekonstrueerd."),
nl, invoer_ruimten.

invoer_ruimten:-
nl, write("Voer ruimtenaam in: "),
read(A),
write("Breedte van "),write(A),write(": "),
read(B),
write("Lengte van "), write(A),write(": "),
read(C),
write("Laag waarop "), write(A),write(" ligt: "),
read(D),
assert(ruimte(A,B,C,D)),
nl, write("Volgende ruimte invoeren: typ r."),
nl, write("Voorwaarden invoeren: typ vw."),
nl, read(E),
((E=r, invoer_ruimten);
 (E=vw,invoer_vw)).

invoer_vw:-
nl, write("Voer ruimtenaam 1 in: "),
read(A),
write("Voer ruimtenaam 2 in: "),
read(B),
write("Voer voorwaarde in (typ bij versie 1 alleen a): "),
read(C),
assert(vw(C,A,B)),
nl, write("Volgende voorwaarde invoeren: typ vw."),
nl, write("Informatie bouwlaag invoeren: typ b."),
nl, read(D),
((D=vw, invoer_vw);
 (D=b, invoer_laag)).

invoer_laag:-
nl, write("Voer bouwlaagnummer in: "),
read(A),
write("Voer bouwhoogte van laag "),write(A),write(" in: "),
read(B),
```

```
write("Voer peilhoogte (niveau) van vloer bouwlaag "),write(A),write(" in: "),
read(C),
assert(laag(A,B,C)),
nl, write("Volgende laag invoeren: typ l."),
nl, write("Klaar met invoer: typ k."),
nl, read(D),
((D=l, invoer_laag);
 (D=k)).
```

"invoer_ruimten" zorgt ervoor dat de gebruiker de ruimtenaam, de ruimteafmetingen en de ruimtehoogte kan invoeren. Het predicaat "read" leest een teken (of meerdere tekens) van het toetsenbord. Na de invoer van een teken of meerdere tekens dient een punt en een return te worden gegeven. Na deze invoer is het predicaat "read" waar en wordt het volgende predicaat genomen. Door "assert" wordt een fact (namelijk de dan bekende ruimte met afmetingen) in de database gezet. De gebruiker kan nu een andere ruimte invoeren of beginnen met voorwaarden in te voeren door een "r" of een "vw" in te typen. Als de invoer "r" is wordt opnieuw geprobeerd het predicaat "invoer_ruimten" waar te maken. Als de invoer "vw" is zal "invoer_vw" verwerkt worden. Op het einde staat een verhindere backtracking (uitroepteken) om te voorkomen dat bij backtracking opnieuw read en write commando's worden uitgevoerd. Voor een uitvoerige beschrijving van de verhinderde backtracking wordt verwezen naar de bijlage "T8-T9-verslagen".

"invoer_vw" werkt op een soortgelijke wijze als "invoer_ruimten". Nu wordt door "assert" een fact ingevoerd die aangeeft dat twee ruimten bij elkaar aansluiten (bijvoorbeeld vw(a,b,c) betekent dat er een voorwaarde a is (a is een afkorting voor aansluitend) die van toepassing is op de ruimten b en c. De routine "invoer_vw" kan opnieuw aangeroepen worden door "vw" te typen. Bij het typen van "k" wordt de clause "ruimtelijk_ontwerp" aangeroepen. Op het einde van de clause "invoer_ruimte" staat een verhinderde backtracking gestoeld op dezelfde redenen als bij de clause "invoer_ruimten". Op dezelfde wijze werkt "invoer_laag".

```
rol1:-
nl,
write("Ruimtelijk ontwerp eerste laag"),
ruimtelijk_ontwerp,!.

zon(N):-
nl,
write("Zoneren van laag "),
write(N),
nl,
zoneer(N),!.

kon(N):-
nl,
write("Konstrueren van laag "),
```

```
write(N),
nl,
konstrueer(N).

ro(N):-
nl,
write("Ruimtelijk ontwerpen van laag "),
write(N),
nl,
ko_lagen(N),!.

onderzoek(B):-
bagof(A,laag2(A),LijstA),
pak3(B,LijstA,LijstB),
hoogste(B,LijstB).

hoogste(C,[]).

hoogste(B,LijstB):-
pak2(C,LijstB,LijstC),
B>=C,
hoogste(C,LijstC).

laag2(A):-laag(A,_,_).
```

"roll" doet het werk van het vroegere programma BLOK25 alleen nu op een geavanceerdere wijze (zie de bijlage "T8-T9-verslagen). Vervolgens zorgt "kon" en "zon" ervoor dat alle ruimtelijk ontwerpen behandeld worden door van elk ruimtelijk ontwerp de konstruktieve mogelijkheden tot invulling in de database te plaatsen. De eindontwerpen, een lijst van ruimten en een lijst van konstruktie-elementen, worden met behulp van het predicaat "assert" in de database van P2W3 geplaatst. Na het ontwerpproces wordt "zon" aangeroepen die de ruimtelijk ontwerpen transformeert tot gezoneerde ontwerpen. "kon" maakt het konstruktief ontwerp van de gezoneerde ontwerpen.

Alle voorgaande handelingen gebeuren per laag, dat moet duidelijk zijn. Eerst zal dus het ruimtelijk ontwerp met "roll" voor de eerste laag gemaakt worden. Vervolgens wordt deze eerste laag gezoneerd en gekonstrueerd met "zon" en "kon". Hierna wordt, bij bijvoorbeeld 3 nog te plaatsen lagen, telkens een laag eerst ruimtelijk opgebouwd, vervolgens gezoneerd en gekonstrueerd en hierna wordt de volgende laag aangepakt. Hiervoor is "cycle". Met "onderzoek" wordt bekeken hoeveel lagen er te plaatsen zijn. Vervolgens wordt met "count", een bekend predicaat, alle lagen afgegaan, gesubstitueerd in "cycle2", die een laag eerst ruimtelijk ontwerpt, daarna zoneert en als laatst konstrueert.

"rol" zet tekst op het scherm en roept het predicaat "ruimtelijk_ontwerp" aan. Evenzo werken "zon", "kon", en "tek". "hoogste" is een hulppredicaat voor "onderzoek" en is verder duidelijk.

```
ruimtelijk_ontwerp:-
bagof(P,ruimte2(P),Olijst),verwerk(Olijst),maaklijst,fail,!.

ruimtelijk_ontwerp.
```

Bij "ruimtelijk_ontwerp" wordt met "bagof" een lijst Olijst gemaakt die alle ruimten bevat. "ruimte2" zorgt ervoor dat alleen de eerste variabele van het predicaat "ruimte" in beschouwing wordt genomen: dit is een programmeerdetail en verder niet van belang. "ruimte2" is een hulpclause en staat als zodanig ook achteraan bij de programmacode. De lijst Olijst wordt verwerkt met de clause "verwerk": dit verwerken houdt in dat ruimtelijke ontwerpen worden geproduceerd door posities van ruimten in de database te stoppen met het predicaat "pos". Door "maaklijst" worden deze posities weer bijeengebracht in een lijst die een ruimtelijk ontwerp bevat: de lijst in het predicaat "ro". "fail" zorgt ervoor dat het predicaat "ruimtelijk_ontwerp" uiteindelijk toch faalt zodat de backtracking van Prolog geactiveerd wordt. Alle mogelijke manieren om de lijst Olijst te verwerken worden onderzocht en ondertussen worden juiste ruimtelijke ontwerpen toch opgeslagen omdat "maaklijst" aan de "fail" voorafgaat. Na de "fail" volgt een verhinderde backtracking omdat als alle mogelijkheden om de lijst Olijst te bewerken gevonden zijn het geen zin heeft na enige tijd weer te gaan backtracken. Als alle mogelijkheden gevonden zijn zal de interpreter naar de volgende regel gaan: "ruimtelijk_ontwerp", en het predicaat "ruimtelijk_ontwerp" zal dus toch slagen. .

```
verwerk([]).

verwerk(Olijst):-
not(pos(B,PosB)),pak2(X,Olijst,Nlijst),
laag(1,GG,GGG),
assert(pos(X,[20,20,GGG])),!,verwerk(Nlijst).

verwerk(Olijst):-
pak(X,Olijst,Nlijst), pos(B,PosB),
voorwaarde_univers(a,X,B),
setof(T,prima(T,X),LijstT),
vorm_goed_lijsten(X,LijstT),
setof(PosC,goedpos(PosC),Lijst),
wisspecialelijst,!,
pick(PosX,Lijst), wislijst(Olijst), niet_overlap(X,PosX),
refresh(X,PosX), verwerk(Nlijst), not(marker(stop)).
```

Nu wordt gekeken naar hoe het verwerken van de lijst Olijst gaat. Eerst wordt met "pak" een ruimte uit de Olijst genomen. De lijst die overblijft wordt Nlijst genoemd. Het predicaat "pak" is een hulppredicaat dat uitgebreid behandeld is in de bijlage "T8-T9-verslagen" en het predicaat staat achteraan in de programmacode. Verder wordt gekeken naar reeds geplaatste ruimten die herkenbaar zijn aan een in de database aanwezig predicaat "pos". De eerste keer dat "verwerk" toegepast wordt zijn er nog geen ruimten geplaatst en zal dus iets anders gedaan moeten worden. Stel nu dat al een ruimte geplaatst is en "pos" dus

waar wordt. Dan wordt gezocht naar een voorwaarde die aangeeft dat de uit de lijst gepakte ruimte aangesloten moet liggen bij de ruimte die al geplaatst is. Is voorgaande voorwaarde onvindbaar dan zal backtracking eerst proberen "pos" op een andere manier waar te maken (dus een andere reeds geplaatste ruimte te zoeken) en mocht dit niet baten dan zal een andere ruimte uit de lijst genomen worden. Door een ruimte uiteindelijk dus altijd zo uit de lijst te nemen dat hij moet aansluiten bij een bestaande geplaatste ruimte worden de zoekmogelijkheden sterk beperkt. "setof" vindt hierna alle geplaatste ruimten die een voorwaarde met de uit de lijst geplaatste ruimte hebben en zet deze ruimten in lijst Tlijst. "vorm_goed_lijsten" zorgt ervoor dat in de database predicaten "specialelijst" ontstaan die als variabele een lijst hebben die posities bevat waar de uit de lijst genomen ruimte zou kunnen worden geplaatst. Ieder feit "specialelijst" bevat te plaatsen posities afhankelijk van 1 reeds geplaatste ruimte. De intersection van de verschillende lijsten onder "specialelijst" zullen dus de werkelijk te plaatsen punten zijn. De intersection van deze punten worden uitgevoerd met "setof" en hierna worden alle feiten "specialelijst" uit de database gewist door het predicaat "wisspecialelijst". Nu een lijst met goede posities voor het te plaatsen blok is gevonden in de lijst Lijst kan hier de eerste positie voor genomen worden door het predicaat "pick" waarna alle oude feiten "pos" uit de database gewist worden door het predicaat "wislijst". Nu moet nog gekeken worden of de ruimte geen andere ruimten overlapt door het predicaat "niet_overlap". Als de gekozen positie van de ruimte al deze testen heeft overleeft kan de positie van de ruimte in de database plaatsen worden met "refresh". Nu moet de lijst Nlijst nog verwerkt worden met "verwerk". De lijst Nlijst was de lijst Olijst waaruit het nu geplaatste blok werd weggenomen.

```
vorm_goed_lijsten(X,Lijst):-pak2(T,Lijst,Nlijst),
maak_goed_lijst(T,X,Lijst2),assert(specialelijst(Lijst2)),
vorm_goed_lijsten(X,Nlijst).
```

"vorm_goed_lijsten(X,LijstT)" zorgt bij invoer van een te plaatsen ruimte X een lijst, van ruimten die een voorwaarde gemeen hebben met deze te plaatsen ruimte, LijstT, dat in de database de predicaten "specialelijst" gevormd worden. Eerst wordt door "pak2" het eerste element uit de lijst LijstT genomen en de overgebleven lijst wordt Nlijst. Dit genomen element is een ruimtenaam. Met "maak_goed_lijst" wordt een lijst Lijst2 gemaakt die alle posities bevat waar de te plaatsen ruimte zich kan bevinden als aan de voorwaarde wordt voldaan die de te plaatsen ruimte verbindt met de uit de LijstT genomen ruimte. Door "assert" wordt deze lijst in de database ingevoerd. Nu wordt "vorm_goed_lijst" toegepast op de te plaatsen ruimte en Nlijst, de lijst die overbleef nadat de in beschouwing genomen geplaatste ruimte uit de LijstT genomen werd. Als de lijst leeg is kan "vorm_goed_lijsten(X,[])" waargemaakt worden en kan binnen de clause "verwerk" verder gegaan worden met de behandeling "setof". Deze behandeling vond de posities die in alle lijsten van het predicaat of fact "specialelijst" staan. Een positie staat in alle lijsten als hij voldoet in het predicaat "goedpos".

```
vorm_goed_lijsten(X,[]).
```

```
vorm_goed_lijsten(X,Lijst):-pak2(T,Lijst,Nlijst),
maak_goed_lijst(T,X,Lijst2),assert(specialelijst(Lijst2)),
vorm_goed_lijsten(X,Nlijst).
```

"prima" dient om alle geplaatste ruimten te vinden die door middel van een voorwaarde met het te plaatsen blok samenhangen. "pos" vindt geplaatste blokken en met "voorwaarde_univers" worden de voorwaarden gevonden waarmee het geplaatste blok samenhangt met het te plaatsen blok.

```
goedpos(Pos):-bagof(Lijst,specialelijst(Lijst),Metalijst),
pak2(Lijst2,Metalijst,NMetalijst),
pick(Pos,Lijst2), check(Pos,NMetalijst).
```

"goedpos" maakt eerst van alle lijsten van "specialelijst" een metalijst Metalijst door "bagof". De standaardpredicaten "bagof" en "setof" zijn vergelijkbaar met het in de bijlage "T8-T9-verslagen" behandelde standaardpredicaat "findall". Uit deze lijst Metalijst wordt het eerste element genomen, een lijst, met "pak2", en uit deze lijst wordt met "pick" een positie genomen. Er wordt gecontroleerd of deze positie ook in de andere lijsten zit door "check". Als "check" waar wordt zit de positie dus is alle lijsten waarmee "goedpos" waar wordt.

```
check(Pos,[]).

check(Pos,NMetalijst):-
pak2(Lijst,NMetalijst,NNMetalijst),
pick(Pos,Lijst),!, check(Pos,NNMetalijst).
```

"check" neemt met "pak2" een lijst Lijst en controleert of de positie in deze lijst zit door "pick". Als dit zo is wordt backtracking verhinderd en wordt gekeken of de positie ook in de rest van de lijsten staat door "check". Blijft uiteindelijke een lege lijst over dan zit de positie in alle lijsten en moet "check" dus waar worden. Dit gebeurt met behulp van "check(Pos,[]).

```
maaklijst:-bagof([X,P],pos(X,P),Lijst),
interactief(Lijst).

interactief(Lijst):-
wcreategraphics(plattegrond,255,255,255),
teken_tussenstand(Lijst),
nl,
nl,write('Ontwerp niet meenemen naar volgende ontwerpstap: a.'),
nl,write('Ontwerp meenemen naar volgende ontwerpstap: b.'),
nl,write('Ontwerp aanpassen en daarna meenemen naar volgende ontwerpstap: c.'),
nl,write('Alle nu nog komende ontwerpen inclusief deze overslaan en door naar volgende ontwerpstap: d.'),
nl,nl, write('Uw keuze: '),
```

```
read(A),!,
((A=d, assert(marker(stop)));
 (A=b, assert(ontwerp(Lijst,[],[])));
 (A=a);
 (A=c, interactief2(Lijst,Lijst2), assert(ontwerp(Lijst2,[],[])))),
wdeletegraphics(plattegrond),!.

interactief2(Lijst,Lijst2):-
etaleer(Lijst),
nl,
nl,write('Het wijzigen van een ontwerp kan op grond van flexibiliteit zonder'),
nl,write('rekening te houden met de voorwaarden bij de invoer. Let goed op dat ruimten'),
nl,write('elkaar bijvoorbeeld kunnen overlappen of los van elkaar kunnen liggen.'),
nl,
nl,write('Welke ruimte wilt U wijzigen van plaats, geef alleen de naam van de ruimte'),
nl,write('gevolgd door een punt en een return: '),
read(A),
pak([A,[P,Q,R]],Lijst,Tussenlijst),
nl,
nl,write('Geef de x-waarde van de nieuwe positie van ruimte '), write(A), write(': '),
read(B),
nl,write('Geef de y-waarde van de nieuwe positie: '),
read(C),
voegbij([A,[B,C,R]],Tussenlijst,Lijst2),
wdeletegraphics(plattegrond),
wcreategraphics(plattegrond),
teken_tussenstand(Lijst2),
nl,nl,write('Gezien, toets willekeurig teken, punt en return: '),
read(QQQ).

etaleer([]).

etaleer(Lijst):-
pak2(Ruimte,Lijst,Lijst2),
pak2(Naam,Ruimte,Rest),
nl, write(Ruimte),
etaleer(Lijst2).

maaklijst.
```

Bij "ruimtelijk_ontwerp" is te zien dat als "verwerk" geslaagd is, "maaklijst" geactiveerd wordt. "maaklijst" vindt alle posities in de database door te zoeken naar de feiten "pos" in de database. Deze gevonden feiten worden door "bagof" in de lijst Lijst gezet. Deze lijst wordt in de database gezet met "assert". Als er geen gevonden posities zijn zal "bagof" niet slagen maar er is geen enkele reden waarom "maaklijst" zou moeten falen. Vandaar dat "maaklijst" ook als fact vermeld staat. "maaklijst" roept ook "interactief" aan. "interactief" roept een grafische window aan door het P2W3-systeempredicaat "wcreategraphics" met de naam "plattegrond". Hierna wordt wat tekst op het scherm gezet en wordt een eenvoudige plattegrond getekend met "teken_tussenstand". De gebruiker kan kiezen uit

enkele opties. Het ontwerp wordt al dan niet in de database geplaatst door "assert". Wil de gebruiker stoppen met deze ontwerpstap dan wordt een markeerteken in de database gezet door "assert". Als de gebruiker een ontwerp wil veranderen dan wordt "interactief2" aangeroepen, dat door middel van een aantal vragen met "read" de gewijzigde gegevens laat zien met "teken_tussenstand" en veranderd met behulp van "assert". Het predicaat "etaleer" laat de gegevens van de diverse ruimten op het scherm zien.

```
refresh(X,P):-
(retract(pos(X,F)),fail);(assert(pos(X,P))),!.
```

"refresh(X,P)" zal eerst oude posities van ruimten verwijderen uit de database door het systeempredicaat "retract". "fail" wordt hier toegevoegd om te zorgen dat alle oude posities verwijderd worden en niet slechts 1. Uiteindelijk zal dit altijd falen en vervolgens wordt de nieuwe positie P met ruimte X in de database ingevoegd door "assert", eveneens een systeempredicaat.

```
maak_goed_lijst(A,B,Lijst):-
ruimte(A,BrA,HgA,1),
ruimte(B,BrB,HgB,1),
pos(A,PosA),
laag(1,GG,GGG),
PosA=[K,L,GGG],
Kdelta is (BrB+BrA)/2, Ldelta is (HgA+HgB)/2,
bagof([X,Y,GGG],p(X,Y,K,Kdelta,L,Ldelta),Lijst).
```

"maak_goed_lijst(A,B,Lijst)" maakt een lijst Lijst van posities van ruimte B bij een gegeven positie van ruimte A zodat aan alle voorwaarden waarin zowel ruimte A als ruimte B in genoemd zijn. De gegevens van de ruimten A en B zijn vermeld in de database toen de invoer van de gebruiker tot stand kwam. Nu kunnen deze gegevens opgevraagd worden door het predicaat "ruimte" weer waar te laten maken. De positie van ruimte A wordt opgevraagd door het predicaat "pos". De positie wordt hierna ontleedt in een x- en een y-coordinaat. Nu worden twee variabelen Kdelta en Ldelta gemaakt zoals aangegeven in de programmacode en daarna worden met "bagof" alle posities die voldoen aan het predicaat "p" gezocht en deze posities worden in de lijst Lijst gezet. Het predicaat "p" genereert dus alle posities waar ruimte B zich kan bevinden. Hiervoor zijn de lengte en breedte van ruimte A nodig, de positie van A en de lengte en hoogte van B. De afmetingen van ruimte A en B zijn impliciet verwerkt in de variabelen Kdelta en Ldelta.

```
p(X,Y,K,Kdelta,L,Ldelta):-
(X is fix(K-Kdelta), N is 2*Ldelta-1, count(R,N), Y is fix(L-Ldelta+R));
(X is fix(K+Kdelta), N is 2*Ldelta-1, count(R,N), Y is fix(L-Ldelta+R));
(Y is fix(L-Ldelta), N is 2*Kdelta-1, count(R,N), X is fix(K-Kdelta+R));
(Y is fix(L+Ldelta), N is 2*Kdelta-1, count(R,N), X is fix(K-Kdelta+R)).
```

"p(X,Y,K,Kdelta,L,Ldelta)" zal X en Y zo vormen zodat ruimte B aansluit bij ruimta A. Ruimte A had als positie K en L. Er zijn nu vier mogelijkheden: B ligt boven A, B ligt

links van A, B ligt onder A en B ligt rechts van A. Er zal gevalsonderscheiding toegepast worden en er wordt een geval als voorbeeld genomen. Als B boven A ligt zal de y-waarde van de positie van B altijd de y-waarde van de positie van A zijn plus de halve hoogte van ruimte A en de halve hoogte van B (en dat is precies Ldelta). De x-waarde van ruimte B zal liggen tussen de x-waarde van A min de halve breedte van A en de halve breedte van B en de x-waarde van A plus de halve breedte van A en de halve breedte van B (en dat is precies Kdelta). Om in dit geval de x-waarde te laten verlopen wordt het predicaat "count" gebruikt. "count(R,N)" begint de natuurlijke getallen op te sommen bij backtracking tot de waarde N waarna het predicaat zal falen. Dit predicaat is uitvoerig beschreven in de bijlage "T8-T9-verslagen". Er wordt begonnen met de x-waarde van B bij de x-waarde van A min Kdelta + 1. De "+1" is nodig omdat als de x-waarde van B als de x-waarde van ruimte A min Kdelta genomen wordt, de twee rechthoeken elkaar in de punt raken (ga dit eventueel na) en dat betekent dat ze niet bij elkaar aansluiten. Vervolgens wordt de x-waarde telkens met 1 verhoogd todat de x-waarde gelijk is aan de x-waarde van ruimte A plus Kdelta - 1. De "-1" heeft dezelfde toepassingsreden als de "+1". Het predicaat "count" genereert getallen van 0 tot N: laat N nu gelijk zijn aan 2 * Kdelta - 1 dan bestrijkt "count" precies de getallenreeks waarvan de getallen opgeteld worden bij de x-waarde van A min Kdelta. Op soortgelijke wijze kunnen de andere gevallen ook behandeld worden. De gevallen zijn in het predicaat "p" herkenbaar door de scheiding met ";". Het ";"-teken staat voor de logische disjunctie.

```
lijst_vw(A,Lijst_vw):-bagof(p(T,B),goedvw(T,A,B),Lijst_vw).

lijst_vw(_,[]).
```

"lijst_vw(A,Lijst_vw)" maakt een lijst Lijst_vw die alle voorwaarden bevat waarin ruimte A genoemd is. De andere ruimte die in de voorwaarde staat moet wel al geplaatst zijn en hiervoor zorgt het predicaat "goedvw". Wordt niets gevonden dan voldoet een lege lijst omdat in de code "lijst_vw(_,[])" staat. Mocht geen voorwaarde gevonden worden waarin ruimte A genoemd is dan faalt het predicaat "bagof" namelijk.

```
goedvw(T,A,B):-pos(B,C),voorwaarde_univers(T,A,B).

voorwaarde_univers(T,A,B):-vw(T,A,B).

voorwaarde_univers(T,A,B):-vw(T,B,A).
```

"goedvw" zoekt naar posities en vindt dus slechts geplaatste ruimten. Vervolgens moet het predicaat "voorwaarde_univers" voldoen. "voorwaarde_univers" is gemaakt om voorwaarden die bij de gebruikersinvoer op een bepaalde manier zijn ingevoerd ook anders te kunnen interpreteren. Een ruimte A die verbonden is met een ruimte B is hetzelfde als een ruimte B die verbonden in met een ruimte A. De gebruiker voert echter maar een van de voorgenoemde relaties in. Bij de invoerverwerking worden in de database feiten geladen met de naam "vw".

```
voorwaarde(a,A,PosA,B,PosB) :- PosA = [X,Y,Z],PosB =
[P,Q,R],Z=R,ruimte(A,BrA,HgA,_),ruimte(B,BrB,HgB,_),
E1 is (Q-Y), abs(E1,E), F is HgA/2 + HgB/2, G1 is (P-X),
abs(G1,G),
H is BrA/2 + BrB/2,
((E=:=F,G<H);(E<F,G=:=H)),!.

voorwaarde(o,A,_,A,_).

voorwaarde(o,A,PosA,B,PosB) :-PosA = [X,Y,Z],PosB =
[P,Q,R],ruimte(A,BrA,HgA,_),ruimte(B,BrB,HgB,_),D1 is (P-X),
abs(D1,D),
E is BrA/2 + BrB/2,F1 is (Q-Y),abs(F1,F),G is HgA/2 + HgB/2,
D < E, F < G,!.

voorwaarde(no,A,PosA,B,PosB):-
not(voorwaarde(o,A,PosA,B,PosB)),!.

niet_overlap(A,PosA):-bagof(B,pos2(B),Lijst),
kijk_na(A,PosA,Lijst),!.
kijk_na(A,PosA,[]).
kijk_na(A,PosA,Lijst):-pak2(B,Lijst,Nieuwe_lijst),pos(B,PosB),
voorwaarde(no,A,PosA,B,PosB),!,kijk_na(A,PosA,Nieuwe_lijst),!.

wisalles:-retractall(pos(A,B)),retractall(lijst(C)),
retractall(vw(D,E,F)),retractall(object(G,H,I,J)).
```

De predicaten "voorwaarde", "niet_overlap" en "wisalles" worden niet meer gebruikt maar staan voor toekomstige uitbreidingen nog wel in de code.

```
wislijst([]).

wislijst(Nlijst):-pak2(P,Nlijst,NNlijst),((retract(pos(P,_)),
fail);true), wislijst(NNlijst),!.

wisspecialelijst:- retractall(specialelijst(_));true.
```

"wislijst(Nlijst)" wist van alle ruimten die in de Nlijst vermeld staan alle vermelde posities in de database. Eerst wordt een ruimte uit de lijst gepakt door middel van de clause "pak2" en hiervan worden alle posities gewist door het standaarpredicaat "retract". Vervolgens wordt de lijst van ruimten die overblijft na het pakken van de eerste ruimte behandeld door de clause "wislijst".

```
lb(A,[X,Y,R]) :- ruimte(A,Bra,Hga,Laag),plaats(A,[P,Q,R]), X is
fix(P-Bra/2),
Y is fix(Q+Hga/2).

rb(A,[X,Y,R]) :- ruimte(A,Bra,Hga,_), plaats(A,[P,Q,R]), X is
```

```
fix(P+Bra/2),
Y is fix(Q+Hga/2).

ro(A,[X,Y,R]) :- ruimte(A,Bra,Hga,_), plaats(A,[P,Q,R]), X is
fix(P+Bra/2),
Y is fix(Q-Hga/2).

lo(A,[X,Y,R]) :- ruimte(A,Bra,Hga,_), plaats(A,[P,Q,R]), X is
fix(P-Bra/2),
Y is fix(Q-Hga/2).
```

Nu wordt de code om zoneringen te ontwikkelen behandeld. Het is in dit verband nodig te weten hoe bij een ruimte de hoekpunten gevonden kunnen worden als de afmetingen en de positie van de ruimte bekend zijn. Er zijn vier gevallen, c.q. hoekpunten. Met gevalsonderscheiding wordt 1 hoekpunt behandeld waarna de andere hoekpunten op soortgelijke wijze kunnen worden gevonden. "lb(A,[X,Y]" zoekt bij ruimte A de linkerbovenhoek (lb is een afkorting voor linksboven) en geeft de positie als lijst van twee coordinaten X en Y. Door "ruimte" en "plaats" worden de benodigde gegevens gekoppeld aan de variabelen waarna door een simpele rekenkundige bewerking X en Y worden vastgesteld. Voor de hoeken rechtsboven (rb), rechtsonder(ro) en linksonder (lo) gelden dezelfde clauses met een aangepaste rekenkundige bewerking.

```
zoneer(N):-
ontwerp(Lijstro,Lijstzones,Lijstelementen),
voeg_in(Lijstro,N),
pap(Lijstro,N),
lz(Lijstzones2),
z_verwerk(Lijstzones2,[],Sortlijst),
lijstomdraai(Sortlijst,[],Sortlijst2),
append(Lijstzones,Sortlijst2,Lijstzones3),
retract(lz(Lijstzones2)),
assert(ontwerp2(Lijstro,Lijstzones3,Lijstelementen)),
write("*"),
fail.
```

"zoneer" vraagt eerst een ruimtelijk ontwerp op door "ontwerp" waarbij het ruimtelijk ontwerp in de lijst Lijstro staat. "voeg_in" zorgt ervoor dat het ruimtelijk ontwerp wederom in de database gezet wordt (Bij het ruimtelijk ontwerpen wordt telkens de database leeg gemaakt). Het ontwerp wordt gezoneerd door "pap". "z_verwerk" zorgt voor een gesorteerde lijst van zones, zodat het ontwerp goed getekend kan worden". "fail" zorgt nadat het predicaat "pap" waar gemaakt is, dat een ander ruimtelijk ontwerp in de database gezocht wordt omdat "ontwerp" waar gemaakt wordt op een andere manier. Als na een tijd alle ruimtelijke ontwerpen gezoneerd zijn, moet "pap" toch waar zijn zodat we "maak" ook als fact in de code vermelden.

```
pap(Lijstro,N):-
```

```
setof(Hoekpunt,vind_hoekpunten(Hoekpunt),Lijsthoekpunten),
assert(hp(Lijsthoekpunten)),
setof(Rechthoek,vind_rechthoek(Rechthoek),Lijstrechthoeken),
assert(rh(Lijstrechthoeken)),
setof(B_rechthoek,bruikbare_rechthoek(B_rechthoek),
Lijstbruikbarerechthoeken),
assert(br(Lijstbruikbarerechthoeken)),
br(Lijstbruikbarerechthoeken),
maak([],Lijstbruikbarerechthoeken,Lijstro,N),!.
```

"pap" zet eerst in de lijst Lijsthoekpunten alle mogelijke hoekpunten door "setof" waarna deze lijst in de database wordt gezet met "assert". Door het predicaat "vind_hoekpunt" worden de hoekpunten gevonden. Vervolgens worden alle mogelijke rechthoeken gezocht die uit de hoekpunten gemaakt kunnen worden met de volgende "setof": het hier van belang zijnde predicaat is "vind_rechthoek". De lijst Lijstrechthoeken die de rechthoeken bevat wordt in de database gezet door "assert". Door "setof" en "bruikbare_rechthoek" worden de rechthoeken geselecteerd die werkelijk een waarde hebben voor het konstruktief ontwerp. In dit geval selecteren we rechthoeken die geen andere ruimten doorsnijden en die langs de gehele rand ruimten bevat. Een lijst van bruikbare rechthoeken, de rechthoeken die werkelijk een waarde hebben, wordt in de database opgeslagen door middel van het predicaat "br". Met "maak" wordt het ruimtelijk ontwerp als een puzzel volgelegd met de bruikbare rechthoeken.

```
zoneer(N):-
retractall(plaats(_,_)),
retractall(hp(_)),
retractall(br(_)),
retractall(rh(_)),
retractall(ontwerp(_,_,_)).

voeg_in(Lijst,N):-
retractall(plaats(_,_)),
retractall(hp(_)),
retractall(br(_)),
retractall(rh(_)),
voeg_in2(Lijst,N),!.

voeg_in2([],N).

voeg_in2(Lijst,N):-
pak2([A,[P,Q,R]],Lijst,Nlijst),
ruimte(A,B,C,D),
((N=D,assert(plaats(A,[P,Q,R])));(not(N=D))),
voeg_in2(Nlijst,N).
```

Als "zoneer" niet meer anders waargemaakt kan worden, wist deze eerst de database met de predicaten "plaats", "hp", "br" en "rh" zodat bij het consulteren van een informatielijst

in deze predicaten niet per ongeluk de verkeerde informatie wordt gelezen. Er is op 1 moment altijd van elk predicaat maar 1 vorm gewenst. "voeg_in2" wordt aangeroepen die in de database op wederom recursieve wijze de posities van ruimten invoert van het op dat moment relevante ruimtelijke ontwerp. Als herhaling kan gezegd worden dat de recursieve manier van werken inhoudt dat het eerste element van een lijst genomen wordt en bewerkt waarna we de lijst die overblijft bewerken (er weer een element afnemen en bewerken) en hiermee doorgaan totdat de lijst leeg is.

```
vind_hoekpunten([X,Y,Z]) :- ruimte(A,Bra,Hga,Laag),
(lb(A,[X,Y,Z]);rb(A,[X,Y,Z]);ro(A,[X,Y,Z]);lo(A,[X,Y,Z])).
```

"vind_hoekpunt" zoekt een ruimte met "ruimte" en accepteert daarna een linkerbovenhoek, rechterbovenhoek, rechteronderhoek en linkeronderhoek als hoekpunt. Door "setof" op "vind_hoekpunt" los te laten worden dus alle hoekpunten gevonden.

```
vind_rechthoek([X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4]) :-
hp(Lijsthoekpunten),
pak([X1,Y1,Z1],Lijsthoekpunten,Lijsthoekpunten2),
pak([X2,Y2,Z2],Lijsthoekpunten2,Lijsthoekpunten3),
Y2=Y1, X2>X1,
pak([X3,Y3,Z3],Lijsthoekpunten3,Lijsthoekpunten4),
X3=X2, Y3<Y2,
pak([X4,Y4,Z4],Lijsthoekpunten4,Lijsthoekpunten5),
X4=X1, Y3=Y4.
```

"vind_rechthoek" vindt rechthoeken. We nemen een lijst met hoekpunten door "hp" uit de database en nemen een punt hieruit met "pak". Vervolgens nemen we nog een punt en kijken of de y-waarden van de twee punten gelijk zijn en de x-waarde van het tweede punt groter is dan de x-waarde van het eerste punt. Mocht dat niet het geval zijn dan wordt automatisch door backtracking een ander punt genomen. De x-waarde van het tweede punt stellen we groter dan de x-waarde van het eerste punt om dubbele oplossingen te voorkomen. Dit is duidelijk te maken door een linkerbovenhoek van een rechthoek voor te stellen als eerste punt. Als we later de rechteronderhoek van een rechthoek als eerste punt nemen zouden we deze rechthoek twee keer vinden. In totaliteit vinden we elke rechthoek dus twee keer, met slechts de volgorde van punten verschillend.

```
bruikbare_rechthoek([[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],LijstB]):-
rh(Lijstrechthoeken),
pak([X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Lijstrechthoeken,_),
lb(A,[X1,Y1,Z1]),
retractall(langs(RR)),
zoek_rb(A,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4),
setof(B,langs(B),LijstB).
```

Het gecompliceerste predicaat is het predicaat "bruikbare_rechthoek". Hoe is bekend of een rechthoek langs de randen altijd ruimten bevat en geen andere ruimten doorsnijdt?

Daartoe wordt een turtle (schildpad) voorgesteld. Het eerste punt (dat is altijd linksboven) van de rechthoek wordt genomen en en wordt gekeken bij welke ruimte dit punt behoort. Vervolgens loopt de schildpad naar rechts (de rechterbovenhoek van de ruimte wordt gezocht). Deze rechterbovenhoek van deze ruimte moet nu of de linkerbovenhoek van een andere ruimte zijn of het rechterbovenhoekpunt van de rechthoek! In het eerste geval loopt de schildpad weer naar rechts naar de rechterbovenhoek van de ruimte, anders loopt de schildpad naar onder (hij loopt nu langs de rechter verticale zijde van de rechthoek). Dit proces wordt voortgezet rondom de gehele rechthoek. Gaat nergens iets mis dan is er sprake van een bruikbare rechthoek. Gaat wel ergens iets mis, bijvoorbeeld is de rechterbovenhoek van de eerste ruimte niet de rechterbovenhoek van de rechthoek en niet de linkerbovenhoek van een andere ruimte, dan mag geconcludeerd worden dat de rechthoek andere ruimten doorsnijdt of langs de rand soms geen ruimten bevat. Deze schildpadfunctie wordt gemaakt met het predicaat "zoek" waarbij achter "zoek" een liggend streepje komt gevolgd door de richting waarin gezocht wordt. Eerst wordt met "rh" bij de clause "bruikbare_rechthoek" een lijst met rechthoeken genomen. Hieruit wordt een rechthoek genomen met "pak" en er wordt gezocht naar de ruimte met de eerste twee coordinaten van de rechthoek als coordinaten van de linkerbovenhoek. Alle predicaten "langs" in de database die een rol spelen bij het later kunnen vertellen welke ruimten in de rechthoek zitten worden gewist en er wordt gezocht met "zoek_rb". Als "zoek_rb" slaagt is er sprake van een bruikbare rechthoek. Alle ruimten die zijn gepasseerd zijn in de database opgenomen door middel van het predicaat "langs". Met "setof" worden deze ruimten in de lijst LijstB gezet opdat de twee variabelen in "bruikbare_rechthoek", te weten [X1,Y1,X2,Y2,X3,Y3,X4,Y4] en LijstB samen vertellen wat een bruikbare rechthoek is en welke ruimten deze bevat.

```
zoek_rb(A,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4):-
assert(langs(A)),
(aansluitend_rb(A,B), rb(B,[P,Q,R]), P=\=X2, Q=\=Y2,
zoek_rb(B,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4));
(aansluitend_rb(A,B), rb(B,[P,Q,R]), P=:=X2, Q=:=Y2,
zoek_ro(B,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4));
(rb(A,[P,Q,R]), P=:=X2, Q=:=Y2,
zoek_ro(A,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4)).

zoek_ro(A,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4):-
assert(langs(A)),
(aansluitend_ro(A,B), ro(B,[P,Q,R]), P=\=X3, Q=\=Y3,
zoek_ro(B,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4));
(aansluitend_ro(A,B), ro(B,[P,Q,R]), P=:=X3, Q=:=Y3,
zoek_lo(B,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4));
(ro(A,[P,Q,R]), P=:=X3, Q=:=Y3,
zoek_lo(A,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4)).

zoek_lo(A,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4):-
assert(langs(A)),
(aansluitend_lo(A,B), lo(B,[P,Q,R]), P=\=X4, Q=\=Y4,
```

```
zoek_lo(B,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4));
(aansluitend_lo(A,B), lo(B,[P,Q,R]), P=:=X4, Q=:=Y4,
zoek_lb(B,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4));
(lo(A,[P,Q,R]), P=:=X4, Q=:=Y4,
zoek_lb(A,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4)).

zoek_lb(A,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4):-
assert(langs(A)),
(aansluitend_lb(A,B), lb(B,[P,Q,R]), P=\=X1, Q=\=Y1,
zoek_lb(B,X1,Y1,Z1,X2,Y2,X3,Y3,Z3,X4,Y4,Z4));
(aansluitend_lb(A,B), lb(B,[P,Q,R]), P=:=X1, Q=:=Y1);
(lb(A,[P,Q,R]), P=:=X1, Q=:=Y1).
```

"zoek_rb" functioneert hetzelfde als "zoek_ro", "zoek_lo" en "zoek_lb" en er wordt volstaan met de bespreking van het eerstgenoemde predicaat. De eerste variabele bij het predicaat (in de code aangegeven met A) geeft de huidige ruimte aan waar de al genoemde schildpad loopt en het is dus vanzelfsprekend dat deze ruimte al in de database geplaatst wordt met "assert". Nu wordt gekeken of rechts van deze ruimte een andere ruimte ligt zodat de rechterbovenhoek van de eerste ruimte de linkerbovenhoek van de tweede ruimte is met behulp van het predicaat "aansluitend". Is dit het geval dan kan het rechterbovenhoekpunt van de tweede ruimte bepaald worden met "rb". Als de positie van dit punt niet gelijk is aan het rechterbovenhoekpunt van de rechthoek (X2,Y2) dan kan verder gezocht worden naar rechts: "zoek_rb" wordt aangeroepen maar nu met de schildpad op de tweede ruimte. Als de positie van dit punt gelijk is aan het rechterbovenhoekpunt van de rechthoek (X2,Y2) dan kan verder gezocht worden naar onder: "zoek_ro" wordt aangeroepen maar nu met de schildpad op de tweede ruimte. Als naast de eerste ruimte geen tweede ruimte zit dan moet het rechterbovenpunt van de eerste ruimte wel de rechterbovenpunt van de rechthoek zijn: er wordt dan ook naar onder gezocht met de schildpad op de eerste ruimte. Voldoen een van deze drie voorwaarden niet dan mag het predicaat "zoek" falen: er is dan immers een doorsnijding of een lege ruimte binnen de rechthoek. Door met "setof" weer "over" "bruikbare_rechthoek" te gaan worden alle bruikbare rechthoeken gevonden en dat gebeurt bij de clause "konstrueer1".

```
aansluitend_rb(A,B):-
rb(A,[X,Y,R]),
lb(B,[X,Y,R]).

aansluitend_lb(A,B):-
lb(A,[X,Y,R]),
lo(B,[X,Y,R]).

aansluitend_lo(A,B):-
lo(A,[X,Y,R]),
ro(B,[X,Y,R]).

aansluitend_ro(A,B):-
ro(A,[X,Y,R]),
```

```
rb(B,[X,Y,R]).
```

"aansluitend" functioneert voor alle 4 de gevallen op soortgelijke wijze en hier wordt volstaan met de behandeling van 1 geval. "aansluitend_rb(A,B)" geeft aan dat ruimte B aan ruimte A ligt op zodanige wijze dat de bovenste horizontale lijnen van de ruimten gelijk lopen. Om dit te bereiken moet de rechterbovenhoek van A de linkerbovenhoek van B zijn: door bij "rb" en "lb" dezelfde variabelen voor de posities te gebruiken wordt Prolog geforceerd hier de juiste ruimten te kiezen mits deze bestaan.

```
maak(Lijstmetzones,Lijstbruikbarerechthoeken,Lijstro,N):-
(goed(Lijstmetzones,N),not(fout(Lijstmetzones)),
assert(lz(Lijstmetzones)),!);
(pak3(Bruikbare_rechthoek,Lijstbruikbarerechthoeken,Lijst2),
 test(Bruikbare_rechthoek,Lijstmetzones),
 voegbij2(Bruikbare_rechthoek,Lijstmetzones,Lijstmetzones2),
 maak(Lijstmetzones2,Lijst2,Lijstro,N)),fail.
maak(Lijstmetzones,Lijstbruikbarerechthoeken,Lijstro,N).
```

Een volgend gecompliceerd predicaat is "maak". "maak" zorgt ervoor dat het ruimtelijke ontwerp wordt "volgelegd" met bruikbare rechthoeken zodat het gehele ruimtelijke ontwerp bedekt is en bruikbare rechthoeken elkaar niet overlappen. "maak" kent drie variabelen. De eerste lijst is in beginsel leeg. Hier wordt telkens bij recursie een bruikbare rechthoek bijgestopt als hij voldoet. De tweede variabele bevat de lijst die alle bruikbare rechthoeken bevat. De derde variabele bevat de lijst met het ruimtelijk ontwerp. Deze laatst genoemde lijst speelt bij deze clause geen rol maar het ruimtelijk ontwerp moet zogenaamd "door de predicaten getrokken worden" zodat verderop (bij andere clauses) het wel weer gebruikt kan worden, met andere woorden mag het ruimtelijk ontwerp niet verloren gaan. Eerst wordt bekeken of de lijst met bruikbare rechthoeken, hier ook wel zones genoemd, correct is met behulp van "goed". Als alle ruimten van het ruimtelijk ontwerp reeds bedekt worden door een zone is een lijst met zones of bruikbare rechthoeken geschikt en zal het predicaat "goed" slagen. Hierna wordt bekeken of de lijst met zones niet al is gevonden met alleen de zones in een andere volgorde (dit zou namelijk identieke oplossingen leveren) met het predicaat "fout". De negatie van dit predicaat met "not" zorgt ervoor dat alle gevallen worden onderzocht zodat het nooit voorkomt dat lijsten twee maal voorkomen. Als aan beide voorgaande predicaten is voldaan ("goed" en "not(fout)" kan de lijst opgeslagen worden met "assert". Verhinderde backtracking zorgt ervoor dat niet nodeloos naar andere oplossingen wordt gezocht die er toch niet zijn. Is een lijst met zones nog niet goed, dan wordt met "pak3" een zone uit de lijst genomen en wordt bekeken of er geen ruimten in deze zone zitten die reeds in de lijst met geplaatste zones zitten door middel van "test". Is dat niet het geval dan kan deze zone dus weer bijgevoegd worden bij onze lijst met zones (de eerste variabele van "maak" die eerst een lege lijst bevatte en nu langzaam gevuld raakt) met het predicaat "voegbij". Vervolgens wordt recursief doorgegaan met "maak": als er een nu een goede lijst met zones gemaakt is wordt deze in de database geladen en anders wordt een volgende zone gezocht om bij te

voegen bij de lijst. "fail" wordt gebruikt om alle oplossingen te verkrijgen. Na alle oplossingen gevonden te hebben moet "maak" wel slagen. Dit wordt gedaan door "maak" ook als fact in de code te vermelden.

```
fout(Lijstmetzones):-
lz(Lijstmetzones2),
verwerk(Lijstmetzones,Lijstmetzones2).
```

Om te kijken of een te gebruiken lijst met zones niet reeds al in de database staat, word- "fout" gebruikt met als variabele een lijst die de tot dan geplaatste zones bevat. Een al bestaande lijst met zones door "lz" wordt genomen en vervolgens wordt "verwerk" gebruikt om te kijken of de twee lijsten dezelfde elementen bevatten. Is dat waar dat wordt "fout" waar en "not(fout)" dus onwaar met als gevolg dat bij "maak" de lijst met zones niet in de database geplaatst zal worden.

```
goed(Lijstmetzones,N):-
bagof(A,ruimte3(A,N),Lijsta),
bagof(B,in_zonelijst(Lijstmetzones,B),Lijstb),
verwerk(Lijsta,Lijstb).
```

Om te kijken of een lijst met zones nu alle ruimten al bevat wordt "goed" gebruikt. Eerst wordt op nu bekende wijze een lijst met alle ruimten gemaakt aan de hand van "bagof". Eveneens wordt een lijst met alle ruimten gemaakt die in de lijst met zones staan. Deze twee lijsten zullen nu evenveel en dezelfde elementen moeten bevatten en dat wordt gecontroleerd met "verwerk".

```
verwerk([],Lijst):-Lijst=[].

verwerk(Lijsta,Lijstb):-
pak2(Ruimte,Lijsta,Lijsta2),
pak(Ruimte,Lijstb,Lijstb2),
verwerk(Lijsta2,Lijstb2).
```

"verwerk" werkt met twee lijsten (in de code Lijsta en Lijstb) en neemt van de eerste lijst het eerste element met "pak2" en met "pak" (een predicaat dat op een willekeurige plaats in een lijst iets kan pakken) wordt gekeken of de tweede lijst dit element ook bevat omdat de variabelenaam in beide pakpredicaten hetzelfde is (namelijk Ruimte). Bij "pak2" wordt Ruimte gebonden en "pak" krijgt nu dus als eerste variabele een gebonden variabele. Hierna worden de overgebleven lijsten verwerkt. Als de eerste lijst leeg is (dus alles is doorzocht) moet de tweede lijst ook leeg zijn en vandaar dat in de code staat: "verwerk([],Lijst):-Lijst=[]".

```
test(Zone,[]).

test(Zone,Lijstmetzones):-
Zone=[Head,Tail],
```

```
pak2(Zone2,Lijstmetzones,Lijst2),
Zone2=[Head2,Tail2],
not(gelijke_elementen(Tail,Tail2)),
test(Zone,Lijst2).

gelijke_elementen(Tail,Tail2):-
pak(Ruimte,Tail,_),
pak(Ruimte,Tail2,_).
```

"test" kijkt of een te plaatsen zone geen ruimten bevat die al in de lijst met geplaatste zones staat. Er wordt een zone uit de lijst gepakt met "pak2". De staart van de lijst die de te plaatsen zone representeert en de staart van de lijst die de al geplaatste zone representeert mogen niet dezelfde elementen bevatten. De staart van een lijst die een zone representeert bevat de namelijk de ruimte-namen van ruimten die in de zone zitten. Met het predicaat "gelijke_elementen" wordt bekeken of er geen gelijke elementen in de twee staarten zitten. Voor de definitie van de staart van een lijst wordt naar de bijlage "T8-T9-verslagen" verwezen.

```
z_verwerk(Lijstzones,Nlijst,Sortlijst):-
pak(Zone,Lijstzones,Restlijst),
z_test(Zone,Restlijst),
voegbij(Zone,Nlijst,NNlijst),
z_verwerk(Restlijst,NNlijst,Sortlijst),!.
```

"z_verwerk" zet een lijst met zones om in een gesorteerde lijst met zones. Om de ontwerpen goed te kunnen tekenen, moeten eerst de zones die links en boven geheel vrij liggen getekend worden, daarna de zones die dan weer links en boven vrij liggen, enz. Om technische redenen is het gemakkelijker om de eerst te tekenen zone achteraan de lijst te zetten en de laatst te tekenen zone vooraan de lijst". "z_verwerk" neemt eerst met "pak" een zone en kijkt of deze zone links en boven geheel vrij ligt met "z_test". Hierna wordt deze zone bijgevoegd bij "Nlijst" tot "NNlijst". Nu wordt nog de "Restlijst" verwerkt.

```
z_test(Zone,Restlijst):-
not(kwadrant4(Zone,Restlijst)).
```

"z_test" vertelt dat een zone links en boven vrij ligt als het niet zo is dat er wel nog een zone links of boven de beschouwde zone ligt, te toetsen met "kwadrant4".

```
kwadrant4(Zone,Restlijst):-
Zone = [[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],_],
pak(Zone2,Restlijst,_),
Zone2 = [[X12,Y12,Z12,X22,Y22,Z22,X32,Y32,Z32,X42,Y42,Z42],_],
X12<X2,
Y12>Y3.
```

"kwadrant4" neemt een zone met "pak" en neemt een andere zone uit de "Restlijst". De linker x-waarde van de zone uit de "Restlijst" moet kleiner zijn dan de rechtse x-waarde van de zone. De bovenste y-waarde van de zone uit de "Restlijst" moet kleiner zijn dan de onderste y-waarde van de zone.

```
achteruitpak(H,[H|[]],[]).

achteruitpak(Z,[H|T],[H|T2]):-
achteruitpak(Z,T,T2).
```

"achteruitpak" neemt steeds het achterste element uit een lijst.

```
in_zonelijst([]):-fail.

in_zonelijst(Lijstmetzones,B):-
pak(Zone,Lijstmetzones,_),
Zone=[Head,Tail], pak(B,Tail,_).
```

"in_zonelijst" kijkt welke ruimten in een zone aanwezig zijn door eerst een zone te nemen en uit de staart van de zone de ruimten te nemen die de zone bevat.

```
konstrueer(N):-
ontwerp2(Lijstro,Lijstzones,Lijstelementen),
konstrueer4(Lijstzones,[],Eindlijst,N),
append(Eindlijst,Lijstelementen,Lijstelementen2),
tekening(Lijstro,Lijstelementen2),
gewichts(Lijstelementen2,Staal,0),
gewichtb(Lijstelementen2,Beton,0),
((integer(Staal), Staal2 is Staal);(Staal2 is fix(Staal))),
((integer(Beton), Beton2 is Beton);(Beton2 is fix(Beton))),
nl,
write('In dit ontwerp is circa '),write(Staal2),write(' kg. staal aanwezig.'),
write(' In dit ontwerp is circa  '),write(Beton2),write( ' kg. beton aanwezig.'),
nl,
nl,write('Konstruktief ontwerp meenemen naar volgende ontwerpstap: a.'),
nl,write('Konstruktief ontwerp niet meenemen: b.'),
nl,
nl,write('Uw keuze: '), read(RR),
((RR=a, assert(ontwerp3(Lijstro,Lijstzones,Lijstelementen2)));
 (RR=b)),
wdeletegraphics(ruimtelijk_ontwerp),
wdeletegraphics(ko_ontwerp),
fail.

konstrueer(N):-
retractall(ontwerp2(_,_,_)).
```

"konstrueer" neemt een lijst met zones door "ontwerp2". Deze lijst met zones "legt" precies een ruimtelijk ontwerp vol. Met het predicaat "konstrueer4" wordt deze lijst doorgenomen en werkelijk gekonstrueerd. Met "tekening" wordt het konstruktief ontwerp getekend en met "gbeton" en "gstaal" worden de globale gewichten staal en beton in het ontwerp berekend en vervolgens afgedrukt door "write". De gebruiker kan kiezen om het ontwerp mee te nemen of niet. De eerste variabele van "ontwerp2" geeft het ruimtelijk ontwerp weer, terwijl de tweede en derde variabele de lijst Lijsteindkonstruktie en Lijst-zones representeert. "fail" wordt gebruikt omdat alle ontwerpen onderzocht moeten worden: we willen immers van elk gezoneerd ontwerp een of meerdere konstruktievoor-stellen. Als alle mogelijkheden doorzocht zijn zal "konstrueer" toch moeten slagen: "konstrueer" als feit wordt in de code toegevoegd.

```
konstrueer4([],Beginlijst,Eindlijst,N):-
Eindlijst=Beginlijst.

konstrueer4(Lijstzones,Beginlijst,Eindlijst,N):-
pak2(Zone,Lijstzones,Nlijstzones),
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
laag(N,_,Hoogte),
Hoogte=Z1,
konstrueer5(Zone,Beginlijst,Tussenlijst),
konstrueer4(Nlijstzones,Tussenlijst,Eindlijst,N).

konstrueer4(Lijstzones,Beginlijst,Eindlijst,N):-
pak2(Zone,Lijstzones,Nlijstzones),
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
laag(N,_,Hoogte),
Hoogte=\=Z1,
konstrueer4(Nlijstzones,Beginlijst,Eindlijst,N).
```

"konstrueer4" pakt op de bekende recursieve wijze een zone uit de lijst en konstrueert deze zone met "konstrueer5". Vervolgens wordt de lijst die overblijft na het nemen van een zone aangeboden aan "konstrueer4".

"laag" is een fact dat informatie geeft over te ruimtelagen. De eerste variabele geeft de naam in de vorm van een getal weer van de ruimtelaag. De tweede variabele geeft de hoogte weer van de laag. De derde variabele geeft de peilhoogte aan van de laag, d.w.z. de peilhoogte van bijvoorbeeld de tweede laag de hoogte van de eerste laag is, enz.

Bij "konstrueer5" kijken we naar de hoogte en de breedte van een zone. De breedte van een zone is eenvoudig de x-waarde van de rechterbovenhoek min de x-waarde van de linkerbovenhoek en evenzo is de hoogte van een zone de y-waarde van de linkerbovenhoek min de y-waarde van de linkeronderhoek. Als de breedte gelijk is aan de hoogte wordt de eerste clause "konstrueer5" waar. Er worden kolommen op de hoekpunten van de zone

geplaatst met "assert" en een ruimtelijk vakwerk wordt in de database gerepresenteerd door een lijst met de posities van de hoekpunten.

De beide andere clauses "konstrueer5" zijn op soortgelijke wijze als de eerste clause te begrijpen. Duidelijk is dat deze clauses nog sterke verbetering behoeven: ze zijn nu op een specifiek geval toegespitst.

KONI

Dit programmagedeelte bevat alle "konstrueer5"-predicaten.

```
konstrueer5(Zone,Elementenlijst,Lijst18):-
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
pak2(Ruimte,Tail,Tail2),
ruimte(Ruimte,_,_,Laag),
laag(Laag,Dikte,Hoogte),
Br is X2-X1, Hg is Y1-Y4,
Br=2,
Hg=2,
Zt is Z1+Dikte,
Kolbr is Dikte/20,
Vldikte is Br/20,
voegbij([kolbet,X1,Y1,Z1,Dikte,2,Kolbr],Elementenlijst,Lijst1),
voegbij([kolbet,X2,Y2,Z2,Dikte,3,Kolbr],Lijst1,Lijst2),
voegbij([kolbet,X4,Y4,Z4,Dikte,1,Kolbr],Lijst2,Lijst3),
voegbij([kolbet,X3,Y3,Z3,Dikte,4,Kolbr],Lijst3,Lijst4),
voegbij([rv2,X4,Y4,Zt,Vldikte],Lijst4,Lijst5),
voegbij([k,X1,Y1,Z1,X1,Y1,Zt],Lijst5,Lijst6),
voegbij([k,X2,Y2,Z2,X2,Y2,Zt],Lijst6,Lijst7),
voegbij([k,X3,Y3,Z3,X3,Y3,Zt],Lijst7,Lijst8),
voegbij([k,X4,Y4,Z4,X4,Y4,Zt],Lijst8,Lijst9),
voegbij([rv,X1,Y1,Zt,X2,Y2,Zt,X3,Y3,Zt,X4,Y4,Zt],Lijst9,Lijst10),
KX1 is X1 + Kolbr/2, KY1 is Y1  - Kolbr/2,
KX2 is X2  - Kolbr/2, KY2 is Y2  - Kolbr/2,
KX3 is X3  - Kolbr/2, KY3 is Y3 + Kolbr/2,
KX4 is X4 + Kolbr/2, KY4 is Y4 + Kolbr/2,
voegbij([stramlijn,KX1,KY1,Z1,KX1,KY1,Zt],Lijst10,Lijst11),
voegbij([stramlijn,KX2,KY2,Z2,KX2,KY2,Zt],Lijst11,Lijst12),
voegbij([stramlijn,KX3,KY3,Z3,KX3,KY3,Zt],Lijst12,Lijst13),
voegbij([stramlijn,KX4,KY4,Z4,KX4,KY4,Zt],Lijst13,Lijst14),
voegbij([stramlijn,KX1,KY1,Zt,KX2,KY2,Zt],Lijst14,Lijst15),
voegbij([stramlijn,KX2,KY2,Zt,KX3,KY3,Zt],Lijst15,Lijst16),
voegbij([stramlijn,KX3,KY3,Zt,KX4,KY4,Zt],Lijst16,Lijst17),
voegbij([stramlijn,KX4,KY4,Zt,KX1,KY1,Zt],Lijst17,Lijst18).
```

"konstrueer5" pakt eerst een ruimte uit de zone met "pak" en bekijkt welke hoogte de laag heeft waarin de zone ligt door "laag". De breedte en hoogte van de zone worden getoetst door de variabelen "Br" en "Hg" te bekijken. Wordt hieraan voldaan, dan kan dit predi-

caat uitgewerkt worden. Wordt niet aan deze voorwaarden voldaan, dan wordt naar een volgend "konstrueer5"-predicaat gegaan. Dikten van konstruktie-elementen worden gegenereerd door het toekennen van waarden aan variabelen en hierna wordt telkens een konstruktie-element toegevoegd aan een lijst door "voegbij". Deze konstruktie-elementen kunnen ook stramienlijnen zijn.

```
konstrueer5(Zone,Elementenlijst,Lijst18):-
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
pak2(Ruimte,Tail,Tail2),
ruimte(Ruimte,_,_,Laag),
laag(Laag,Dikte,Hoogte),
Br is X2-X1, Hg is Y1-Y4,
Br=4,
Hg=4,
Zt is Z1+Dikte,
Kolbr is Dikte/20,
Vldikte is Br/20,
voegbij([kolbet,X1,Y1,Z1,Dikte,2,Kolbr],Elementenlijst,Lijst1),
voegbij([kolbet,X2,Y2,Z2,Dikte,3,Kolbr],Lijst1,Lijst2),
voegbij([kolbet,X4,Y4,Z4,Dikte,1,Kolbr],Lijst2,Lijst3),
voegbij([kolbet,X3,Y3,Z3,Dikte,4,Kolbr],Lijst3,Lijst4),
voegbij([rv4,X4,Y4,Zt,Vldikte],Lijst4,Lijst5),
voegbij([k,X1,Y1,Z1,X1,Y1,Zt],Lijst5,Lijst6),
voegbij([k,X2,Y2,Z2,X2,Y2,Zt],Lijst6,Lijst7),
voegbij([k,X3,Y3,Z3,X3,Y3,Zt],Lijst7,Lijst8),
voegbij([k,X4,Y4,Z4,X4,Y4,Zt],Lijst8,Lijst9),
voegbij([rv,X1,Y1,Zt,X2,Y2,Zt,X3,Y3,Zt,X4,Y4,Zt],Lijst9,Lijst10),
KX1 is X1 + Kolbr/2, KY1 is Y1  - Kolbr/2,
KX2 is X2  - Kolbr/2, KY2 is Y2  - Kolbr/2,
KX3 is X3  - Kolbr/2, KY3 is Y3 + Kolbr/2,
KX4 is X4 + Kolbr/2, KY4 is Y4 + Kolbr/2,
voegbij([stramlijn,KX1,KY1,Z1,KX1,KY1,Zt],Lijst10,Lijst11),
voegbij([stramlijn,KX2,KY2,Z2,KX2,KY2,Zt],Lijst11,Lijst12),
voegbij([stramlijn,KX3,KY3,Z3,KX3,KY3,Zt],Lijst12,Lijst13),
voegbij([stramlijn,KX4,KY4,Z4,KX4,KY4,Zt],Lijst13,Lijst14),
voegbij([stramlijn,KX1,KY1,Zt,KX2,KY2,Zt],Lijst14,Lijst15),
voegbij([stramlijn,KX2,KY2,Zt,KX3,KY3,Zt],Lijst15,Lijst16),
voegbij([stramlijn,KX3,KY3,Zt,KX4,KY4,Zt],Lijst16,Lijst17),
voegbij([stramlijn,KX4,KY4,Zt,KX1,KY1,Zt],Lijst17,Lijst18).

konstrueer5(Zone,Elementenlijst,Lijst20):-
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
pak2(Ruimte,Tail,Tail2),
ruimte(Ruimte,_,_,Laag),
laag(Laag,Dikte,Hoogte),
Zt is Z1+Dikte,
Br is X2-X1, Hg is Y1-Y4,
Br=2,Hg=4,
Dischijf is Dikte/20,
```

```
Divloer is Br/20,
voegbij([schijf_y,X4,Y4,Z4,Dikte,Hg,r,Dischijf],Elementenlijst,Lijst1),
voegbij([schijf_y,X3,Y3,Z3,Dikte,Hg,l,Dischijf],Lijst1,Lijst2),
Yv1 is Y2-1,Yv2 is Y2-2,Yv3 is Y2-3,Yv4 is Y2-4,
voegbij([vloerbet_x,X1,Yv1,Zt,2,Divloer],Lijst2,Lijst3),
voegbij([vloerbet_x,X1,Yv2,Zt,2,Divloer],Lijst3,Lijst4),
voegbij([vloerbet_x,X1,Yv3,Zt,2,Divloer],Lijst4,Lijst5),
voegbij([vloerbet_x,X1,Yv4,Zt,2,Divloer],Lijst5,Lijst6),
voegbij([schijf,X4,Y4,Z4,X1,Y1,Z1,X1,Y1,Zt,X4,Y4,Zt],Lijst6,Lijst7),
voegbij([schijf,X3,Y3,Z3,X2,Y2,Z2,X2,Y2,Zt,X3,Y3,Zt],Lijst7,Lijst8),
X1s is X1+Dischijf/2, X2s is X2-Dischijf/2,
voegbij([stramlijn,X1s,Y1,Z1,X1s,Y4,Z4],Lijst8,Lijst9),
voegbij([stramlijn,X1s,Y1,Zt,X1s,Y4,Zt],Lijst9,Lijst10),
voegbij([stramlijn,X1s,Y1,Z1,X1s,Y1,Zt],Lijst10,Lijst11),
voegbij([stramlijn,X1s,Y4,Z4,X1s,Y4,Zt],Lijst11,Lijst12),
voegbij([stramlijn,X2s,Y2,Z2,X2s,Y3,Z3],Lijst12,Lijst13),
voegbij([stramlijn,X2s,Y2,Zt,X2s,Y3,Zt],Lijst13,Lijst14),
voegbij([stramlijn,X2s,Y2,Z2,X2s,Y2,Zt],Lijst14,Lijst15),
voegbij([stramlijn,X2s,Y3,Z3,X2s,Y3,Zt],Lijst15,Lijst16),
Zv is Zt + Divloer/2,
voegbij([stramlijn,X1s,Y1,Zv,X2s,Y1,Zv],Lijst16,Lijst17),
voegbij([stramlijn,X2s,Y2,Zv,X2s,Y3,Zv],Lijst17,Lijst18),
voegbij([stramlijn,X2s,Y3,Zv,X1s,Y4,Zv],Lijst18,Lijst19),
voegbij([stramlijn,X1s,Y4,Zv,X1s,Y1,Zv],Lijst19,Lijst20).

konstrueer5(Zone,Elementenlijst,Lijst20):-
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
pak2(Ruimte,Tail,Tail2),
ruimte(Ruimte,_,_,Laag),
laag(Laag,Dikte,Hoogte),
Zt is Z1+Dikte,
Br is X2-X1, Hg is Y1-Y4,
Br=4,Hg=2,
Dischijf is Dikte/20,
Divloer is Hg/20,
voegbij([schijf_x,X2,Y2,Z2,Dikte,Br,l,Dischijf],Elementenlijst,Lijst1),
voegbij([schijf_x,X3,Y3,Z3,Dikte,Br,r,Dischijf],Lijst1,Lijst2),
Xv1 is X4,Xv2 is X4+1,Xv3 is X4+2,Xv4 is X4+3,
voegbij([vloerbet_y,Xv1,Y4,Zt,2,Divloer],Lijst2,Lijst3),
voegbij([vloerbet_y,Xv2,Y4,Zt,2,Divloer],Lijst3,Lijst4),
voegbij([vloerbet_y,Xv3,Y4,Zt,2,Divloer],Lijst4,Lijst5),
voegbij([vloerbet_y,Xv4,Y4,Zt,2,Divloer],Lijst5,Lijst6),
voegbij([schijf,X1,Y1,Z1,X2,Y2,Z2,X2,Y2,Zt,X1,Y1,Zt],Lijst6,Lijst7),
voegbij([schijf,X4,Y4,Z4,X3,Y3,Z3,X3,Y3,Zt,X4,Y4,Zt],Lijst7,Lijst8),
Y4s is Y4+Dischijf/2, Y1s is Y1-Dischijf/2,
voegbij([stramlijn,X1,Y1s,Z1,X2,Y1s,Z2],Lijst8,Lijst9),
voegbij([stramlijn,X1,Y1s,Zt,X2,Y1s,Zt],Lijst9,Lijst10),
voegbij([stramlijn,X4,Y4s,Z4,X3,Y4s,Z4],Lijst10,Lijst11),
voegbij([stramlijn,X4,Y4s,Zt,X3,Y4s,Zt],Lijst11,Lijst12),
voegbij([stramlijn,X1,Y1s,Z1,X1,Y1s,Zt],Lijst12,Lijst13),
```

```
voegbij([stramlijn,X2,Y1s,Z1,X2,Y1s,Zt],Lijst13,Lijst14),
voegbij([stramlijn,X4,Y4s,Z4,X4,Y4s,Zt],Lijst14,Lijst15),
voegbij([stramlijn,X3,Y4s,Z3,X3,Y4s,Zt],Lijst15,Lijst16),
Zv is Zt + Divloer/2,
voegbij([stramlijn,X1,Y1s,Zv,X2,Y1s,Zv],Lijst16,Lijst17),
voegbij([stramlijn,X2,Y1s,Zv,X3,Y4s,Zv],Lijst17,Lijst18),
voegbij([stramlijn,X3,Y4s,Zv,X4,Y4s,Zv],Lijst18,Lijst19),
voegbij([stramlijn,X4,Y4s,Zv,X1,Y1s,Zv],Lijst19,Lijst20).

konstrueer5(Zone,Elementenlijst,Lijst28):-
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
pak2(Ruimte,Tail,Tail2), ruimte(Ruimte,_,_,Laag),
laag(Laag,Dikte,Hoogte), Zt is Z1+Dikte,
Br is X2-X1, Hg is Y1-Y4, Hg = 2, Br = 4,
Dikol is Dikte/20, Dibalk is Br/20, Divloer is Hg/20,
voegbij([kolstaal,X1,Y1,Z1,Dikte,2,Dikol],Elementenlijst,Lijst1),
voegbij([kolstaal,X4,Y4,Z4,Dikte,1,Dikol],Lijst1,Lijst2),
voegbij([kolstaal,X2,Y2,Z2,Dikte,3,Dikol],Lijst2,Lijst3),
voegbij([kolstaal,X3,Y3,Z3,Dikte,4,Dikol],Lijst3,Lijst4),
voegbij([balkstaal_x,X3,Y3,Zt,Br,r,Dibalk],Lijst4,Lijst5),
voegbij([balkstaal_x,X2,Y2,Zt,Br,l,Dibalk],Lijst5,Lijst6),
Xv2 is X4+1, Xv3 is X4+2,
Xv4 is X4+3,
voegbij([vloerstaal_y,X4,Y4,Zt,Hg,Divloer],Lijst6,Lijst7),
voegbij([vloerstaal_y,Xv2,Y4,Zt,Hg,Divloer],Lijst7,Lijst8),
voegbij([vloerstaal_y,Xv3,Y4,Zt,Hg,Divloer],Lijst8,Lijst9),
voegbij([vloerstaal_y,Xv4,Y4,Zt,Hg,Divloer],Lijst9,Lijst10),
XM is fix((X1+X2)/2),
voegbij([k,X1,Y1,Z1,X1,Y1,Zt],Lijst10,Lijst11),
voegbij([k,X2,Y2,Z2,X2,Y2,Zt],Lijst11,Lijst12),
voegbij([k,X3,Y3,Z3,X3,Y3,Zt],Lijst12,Lijst13),
voegbij([k,X4,Y4,Z4,X4,Y4,Zt],Lijst13,Lijst14),
voegbij([k,XM,Y1,Z1,XM,Y1,Zt],Lijst14,Lijst15),
voegbij([k,XM,Y3,Z3,XM,Y3,Zt],Lijst15,Lijst16),
voegbij([balk,X1,Y1,Zt,X2,Y2,Zt],Lijst16,Lijst17),
voegbij([balk,X4,Y4,Zt,X3,Y3,Zt],Lijst17,Lijst18),
X4s is X4 + Dikol/2, Y4s is Y4 + Dikol/2,
X3s is X3 - Dikol/2, Y1s is Y1 - Dikol/2,
Zkol is Zt - Dibalk,
voegbij([stramlijn,X4s,Y4s,Z4,X4s,Y4s,Zkol],Lijst18,Lijst19),
voegbij([stramlijn,X4s,Y1s,Z4,X4s,Y1s,Zkol],Lijst19,Lijst20),
voegbij([stramlijn,X3s,Y4s,Z4,X3s,Y4s,Zkol],Lijst20,Lijst21),
voegbij([stramlijn,X3s,Y1s,Z4,X3s,Y1s,Zkol],Lijst21,Lijst22),
Zbalk is Zt - Dibalk/2,
voegbij([stramlijn,X4s,Y1s,Zbalk,X3s,Y1s,Zbalk],Lijst22,Lijst23),
voegbij([stramlijn,X4s,Y4s,Zbalk,X3s,Y4s,Zbalk],Lijst23,Lijst24),
Zvloer is Zt - Divloer/2,
voegbij([stramlijn,X4s,Y1s,Zvloer,X4s,Y4s,Zvloer],Lijst24,Lijst25),
voegbij([stramlijn,X4s,Y4s,Zvloer,X3s,Y4s,Zvloer],Lijst25,Lijst26),
voegbij([stramlijn,X4s,Y4s,Zvloer,X4s,Y1s,Zvloer],Lijst26,Lijst27),
```

```
voegbij([stramlijn,X3s,Y4s,Zvloer,X3s,Y1s,Zvloer],Lijst27,Lijst28).

konstrueer5(Zone,Elementenlijst,Nlijst):-
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
pak2(Ruimte,Tail,Tail2),
ruimte(Ruimte,_,_,Laag),
laag(Laag,Hoogte,Peilhoogte),
onderzoek(Q),
Q=Laag, /*geldt alleen voor bovenste lagen*/
Br is X2-X1, Hg is Y1-Y4,
Br<Hg,
Zt is Z1+ Hoogte,
AantalSpantzones is Hg/2,
Ztop is Zt + 1/14 * Br,
Zhalf is Zt + 1/28 * Br,
Zlaag is Zt - 1/14 * Br,
Xvierde is X1 + Br/4,
Xhalf is X1 + Br/2,
Xdrievierde is X2 - Br/4,
reeks(AantalSpantzones,Nieuwelijst,X1,Z1,Zt,Z2,X2,Y1,Xhalf,Ztop,Xvierde,Xdrievierde,Zhalf,Zlaag),
append(Nieuwelijst,Elementenlijst,Nlijst).

reeks(AantalSpantzones,Nieuwelijst,X1,Z1,Zt,Z2,X2,Y1,Xhalf,Ztop,Xvierde,Xdrievierde,Zhalf,Zlaag):-
Aantaltel is AantalSpantzones + 1,
count(N,Aantaltel),
YS is Y1 - (N*2 - 2),
assert(houtlijst([kolomhout,X1,YS,Z1,X1,YS,Zt])),
assert(houtlijst([kolomhout,X2,YS,Z2,X2,YS,Zt])),
assert(houtlijst([kolomhout,X1,YS,Zlaag,X2,YS,Zlaag])),
assert(houtlijst([kolomhout,X1,YS,Zt,Xhalf,YS,Ztop])),
assert(houtlijst([kolomhout,X2,YS,Zt,Xhalf,YS,Ztop])),
assert(houtlijst([kolomhout,Xhalf,YS,Zlaag,Xhalf,YS,Ztop])),
assert(houtlijst([kolomhout,Xhalf,YS,Zlaag,Xvierde,YS,Zhalf])),
assert(houtlijst([kolomhout,Xhalf,YS,Zlaag,Xdrievierde,YS,Zhalf])),
assert(houtlijst([kolomhout,X1,YS,Zlaag,Xvierde,YS,Zhalf])),
assert(houtlijst([kolomhout,X2,YS,Zlaag,Xdrievierde,YS,Zhalf])),
assert(houtlijst([stramlijn,X1,YS,Z1,X1,YS,Zt])),
assert(houtlijst([stramlijn,X2,YS,Z2,X2,YS,Zt])),
assert(houtlijst([stramlijn,X1,YS,Zlaag,X2,YS,Zlaag])),
assert(houtlijst([stramlijn,X1,YS,Zt,Xhalf,YS,Ztop])),
assert(houtlijst([stramlijn,X2,YS,Zt,Xhalf,YS,Ztop])),
assert(houtlijst([stramlijn,Xhalf,YS,Zlaag,Xhalf,YS,Ztop])),
assert(houtlijst([stramlijn,Xhalf,YS,Zlaag,Xvierde,YS,Zhalf])),
assert(houtlijst([stramlijn,Xhalf,YS,Zlaag,Xdrievierde,YS,Zhalf])),
assert(houtlijst([stramlijn,X1,YS,Zlaag,Xvierde,YS,Zhalf])),
assert(houtlijst([stramlijn,X2,YS,Zlaag,Xdrievierde,YS,Zhalf])),
fail.

reeks(AantalSpantzones,Nieuwelijst,X1,Z1,Zt,Z2,X2,Y1,Xhalf,Ztop,Xvierde,Xdrievierde,Zhalf,Zlaag):-
bagof(Elem,houtlijst(Elem),Nieuwelijst),
```

```
retractall(houtlijst(_)).
```

In tegenstelling tot de voorgaande "konstrueer5"-predicaten worden bij de bovenstaande variant geen eisen gesteld aan de afmetingen. Het konstrueren is voor elke afmeting zone mogelijk met dit predicaat. Nu worden geen konstruktie-elementen toegevoegd aan een lijst, maar worden de elementen in de database gezet met "assert" en "reeks", waardoor het mogelijk is meerdere keren een aantal elementen in de database te zetten. Op het einde wordt met "bagof" deze hele verzameling elementen bij elkaar gehaald en gebruikt.

```
konstrueer5(Zone,Elementenlijst,Nlijst):-
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
pak2(Ruimte,Tail,Tail2),
ruimte(Ruimte,_,_,Laag),
laag(Laag,Hoogte,Peilhoogte),
onderzoek(Q),
Q=Laag, /*geldt alleen voor bovenste lagen*/
Br is X2-X1, Hg is Y1-Y4,
Br>Hg,
Zt is Z1+ Hoogte,
AantalSpantzones is Br/2,
Ztop is Zt + 1/14 * Br,
Zhalf is Zt + 1/28 * Br,
Zlaag is Zt - 1/14 * Br,
Yvierde is Y4 + Hg/4,
Yhalf is Y4 + Hg/2,
Ydrievierde is Y1 - Hg/4,
reeks2(AantalSpantzones,Nieuwelijst,X1,Y1,Z1,Zt,Y4,Zlaag,Yhalf,Ztop,Yvierde,Ydrievierde,Zhalf),
append(Nieuwelijst,Elementenlijst,Nlijst).

reeks2(AantalSpantzones,Nieuwelijst,X1,Y1,Z1,Zt,Y4,Zlaag,Yhalf,Ztop,Yvierde,Ydrievierde,Zhalf):-
Aantaltel is AantalSpantzones + 1,
count(N,Aantaltel),
XS is X1 + (N*2 - 2),
assert(houtlijst([kolomhout,XS,Y1,Z1,XS,Y1,Zt])),
assert(houtlijst([kolomhout,XS,Y4,Z1,XS,Y4,Zt])),
assert(houtlijst([kolomhout,XS,Y1,Zlaag,XS,Y4,Zlaag])),
assert(houtlijst([kolomhout,XS,Y1,Zt,XS,Yhalf,Ztop])),
assert(houtlijst([kolomhout,XS,Y4,Zt,XS,Yhalf,Ztop])),
assert(houtlijst([kolomhout,XS,Yhalf,Zlaag,XS,Yhalf,Ztop])),
assert(houtlijst([kolomhout,XS,Yhalf,Zlaag,XS,Yvierde,Zhalf])),
assert(houtlijst([kolomhout,XS,Yhalf,Zlaag,XS,Ydrievierde,Zhalf])),
assert(houtlijst([kolomhout,XS,Y1,Zlaag,XS,Ydrievierde,Zhalf])),
assert(houtlijst([kolomhout,XS,Y4,Zlaag,XS,Yvierde,Zhalf])),
assert(houtlijst([stramlijn,XS,Y1,Z1,XS,Y1,Zt])),
assert(houtlijst([stramlijn,XS,Y4,Z1,XS,Y4,Zt])),
assert(houtlijst([stramlijn,XS,Y1,Zlaag,XS,Y4,Zlaag])),
assert(houtlijst([stramlijn,XS,Y1,Zt,XS,Yhalf,Ztop])),
assert(houtlijst([stramlijn,XS,Y4,Zt,XS,Yhalf,Ztop])),
assert(houtlijst([stramlijn,XS,Yhalf,Zlaag,XS,Yhalf,Ztop])),
```

```
assert(houtlijst([stramlijn,XS,Yhalf,Zlaag,XS,Yvierde,Zhalf])),
assert(houtlijst([stramlijn,XS,Yhalf,Zlaag,XS,Ydrievierde,Zhalf])),
assert(houtlijst([stramlijn,XS,Y1,Zlaag,XS,Ydrievierde,Zhalf])),
assert(houtlijst([stramlijn,XS,Y4,Zlaag,XS,Yvierde,Zhalf])),
fail.

reeks2(AantalSpantzones,Nieuwelijst,X1,Y1,Z1,Zt,Y4,Zlaag,Yhalf,Ztop,Yvierde,Ydrievierde,Zhalf):-
bagof(Elem,houtlijst(Elem),Nieuwelijst),
retractall(houtlijst(_)).

konstrueer5(Zone,Elementenlijst,Lijst28):-
Zone=[[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],Tail],
pak2(Ruimte,Tail,Tail2),
ruimte(Ruimte,_,_,Laag),
laag(Laag,Dikte,Hoogte),
Zt is Z1+Dikte,
Br is X2-X1, Hg is Y1-Y4,
Hg = 4, Br = 2,
Dikol is Dikte/20, Dibalk is Br/20, Divloer is Hg/20,
voegbij([kolstaal,X1,Y1,Z1,Dikte,2,Dikol],Elementenlijst,Lijst1),
voegbij([kolstaal,X4,Y4,Z4,Dikte,1,Dikol],Lijst1,Lijst2),
voegbij([kolstaal,X2,Y2,Z2,Dikte,3,Dikol],Lijst2,Lijst3),
voegbij([kolstaal,X3,Y3,Z3,Dikte,4,Dikol],Lijst3,Lijst4),
voegbij([balkstaal_y,X4,Y4,Zt,Hg,r,Dibalk],Lijst4,Lijst5),
voegbij([balkstaal_y,X3,Y3,Zt,Hg,l,Dibalk],Lijst5,Lijst6),
Yv1 is Y1-1, Yv2 is Y1-2, Yv3 is Y1-3, Yv4 is Y1-4,
voegbij([vloerstaal_x,X4,Yv1,Zt,Br,Divloer],Lijst6,Lijst7),
voegbij([vloerstaal_x,X4,Yv2,Zt,Br,Divloer],Lijst7,Lijst8),
voegbij([vloerstaal_x,X4,Yv3,Zt,Br,Divloer],Lijst8,Lijst9),
voegbij([vloerstaal_x,X4,Yv4,Zt,Br,Divloer],Lijst9,Lijst10),
YM is fix((Y1+Y4)/2),
voegbij([k,X1,Y1,Z1,X1,Y1,Zt],Lijst10,Lijst11),
voegbij([k,X2,Y2,Z2,X2,Y2,Zt],Lijst11,Lijst12),
voegbij([k,X3,Y3,Z3,X3,Y3,Zt],Lijst12,Lijst13),
voegbij([k,X4,Y4,Z4,X4,Y4,Zt],Lijst13,Lijst14),
voegbij([k,X1,YM,Z1,X1,YM,Zt],Lijst14,Lijst15),
voegbij([k,X2,YM,Z2,X2,YM,Zt],Lijst15,Lijst16),
voegbij([balk,X1,Y1,Zt,X4,Y4,Zt],Lijst16,Lijst17),
voegbij([balk,X2,Y2,Zt,X3,Y3,Zt],Lijst17,Lijst18),
X4s is X4 + Dikol/2, Y4s is Y4 + Dikol/2,
X3s is X3 - Dikol/2, Y1s is Y1 - Dikol/2,
Zkol is Zt - Dibalk,
voegbij([stramlijn,X4s,Y4s,Z4,X4s,Y4s,Zkol],Lijst18,Lijst19),
voegbij([stramlijn,X4s,Y1s,Z4,X4s,Y1s,Zkol],Lijst19,Lijst20),
voegbij([stramlijn,X3s,Y4s,Z4,X3s,Y4s,Zkol],Lijst20,Lijst21),
voegbij([stramlijn,X3s,Y1s,Z4,X3s,Y1s,Zkol],Lijst21,Lijst22),
Zbalk is Zt - Dibalk/2,
voegbij([stramlijn,X4s,Y1s,Zbalk,X4s,Y4s,Zbalk],Lijst22,Lijst23),
voegbij([stramlijn,X3s,Y1s,Zbalk,X3s,Y4s,Zbalk],Lijst23,Lijst24),
Zvloer is Zt -  Divloer/2,
```

```
voegbij([stramlijn,X4s,Y1s,Zvloer,X4s,Y4s,Zvloer],Lijst24,Lijst25),
voegbij([stramlijn,X4s,Y4s,Zvloer,X3s,Y4s,Zvloer],Lijst25,Lijst26),
voegbij([stramlijn,X4s,Y4s,Zvloer,X4s,Y1s,Zvloer],Lijst26,Lijst27),
voegbij([stramlijn,X3s,Y4s,Zvloer,X3s,Y1s,Zvloer],Lijst27,Lijst28).

konstrueer5(Zone,Elem,Elem).
```

TEK1

Het programma TEK1 bevat alle tekenroutines voor het programma en de code voor het plaatsen van ruimten op een reeds gekonstrueerde laag, met andere woorden het ruimtelijk ontwerpen van alle lagen behalve laag 0.

```
teken:-
ontwerp3(Ruim_ontwerp,Lijst_zon,Lijst_kon),
tekening(Ruim_ontwerp,Lijst_kon),
toets,
fail.

toets:-
read(A),
wdeletegraphics(ruimtelijk_ontwerp),
wdeletegraphics(ko_ontwerp),!.
```

"teken" maakt eerst kennis met een ontwerp door "ontwerp3", vervolgens wordt "tekening" aangeroepen. Toets dient om een rust tussen de tekeningen te maken.

```
tekening(Ruim_ontwerp,Lijst_kon):-
wcreategraphics(ruimtelijk_ontwerp,255,255,255),
teken_ruim(Ruim_ontwerp),
wcreategraphics(ko_ontwerp,255,255,255),
teken_kon(Lijst_kon),!.
```

"tekening" maakt een grafische window met het P2W3-systeempredicaat "wcreategraphics" en roept daarna "teken_ruim" aan. Als "teken_ruim" waar is, wordt een grafische window aangeroepen waar het konstruktief ontwerp in te zien zal zijn: "ko_ontwerp". Met "teken_kon" wordt de konstruktie getekend.

```
teken_kon([]).

teken_kon(Lijst_kon):-
achteruitpak(Element,Lijst_kon,Lijst_kon2),
teken_kon_elem(Element),
teken_kon(Lijst_kon2),!.

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
```

```
Type=stramlijn,
Rest=[X1,Y1,Z1,X2,Y2,Z2],
t_stramlijn(X1,Y1,Z1,X2,Y2,Z2).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=kolomhout,
Rest=[X1,Y1,Z1,X2,Y2,Z2],
t_kolomhout(X1,Y1,Z1,X2,Y2,Z2).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=vloerbet_x,
Rest=[X2,Yv1,Zt,L,_],
t_betonvloer_x_richt(X2,Yv1,Zt,L).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=vloerbet_y,
Rest=[Xv1,Y4,Zt,L,_],
t_betonvloer_y_richt(Xv1,Y4,Zt,L).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=schijf_x,
Rest=[X3,Y3,Z3,Dikte,Br,Type2,_],
t_schijf_x_richt(X3,Y3,Z3,Dikte,Br,Type2).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=schijf_y,
Rest=[X3,Y3,Z3,Dikte,Br,Type2,_],
t_schijf_y_richt(X3,Y3,Z3,Dikte,Br,Type2).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=kolstaal,
Rest=[X1,Y1,Z1,Dikte,Kwadrant,_],
t_staalkolom(X1,Y1,Z1,Dikte,Kwadrant).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=balkstaal_x,
Rest=[X3,Y3,Zt,Br,Type2,_],
t_staalbalk_x_richt(X3,Y3,Zt,Br,Type2).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=balkstaal_y,
Rest=[X3,Y3,Zt,Br,Type2,_],
```

```
t_staalbalk_y_richt(X3,Y3,Zt,Br,Type2).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=vloerstaal_x,
Rest=[X3,Y3,Zt,Br,_],
t_staalvloer_x_richt(X3,Y3,Zt,Br).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=vloerstaal_y,
Rest=[X3,Y3,Zt,Br,_],
t_staalvloer_y_richt(X3,Y3,Zt,Br).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=kolbet,
Rest=[X3,Y3,Z3,Dikte,Kwadrant,_],
t_betonkolom(X3,Y3,Z3,Dikte,Kwadrant).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=rv2,
Rest=[X3,Y3,Z3,_],
t_rv2(X3,Y3,Z3).

teken_kon_elem(Element):-
pak2(Type,Element,Rest),
Type=rv4,
Rest=[X3,Y3,Z3,_],
t_rv4(X3,Y3,Z3).

teken_kon_elem(Element).
```

Met "teken_kon_elem" wordt van een element het type genomen door "pak". Afhankelijk van het type wordt een bepaalde tekenroutine aangeroepen. Deze tekenroutine's beginnen allemaal met een "t_".

```
t_kolomhout(X1,Y1,Z1,X2,Y2,Z2):-
trans(X1,Y1,Z1,SS1,ST1),
trans(X2,Y2,Z2,SS2,ST2),
wgraph(ko_ontwerp,pen_color(255,255,0)),
wgraph(ko_ontwerp,polyline(SS1-ST1,SS2-ST2)).
```

Een voorbeeld van een tekenroutine is "t_kolomhout". Door de "trans"-predicaten wordt de real-world positie omgezet in schermposities. Met "wgraph", een P2W3-systeempredicaat, wordt vervolgens iets op het scherm getekend, waarbij "pen_color" voor de kleur staat van een lijn en "polyline" een lijn tekent.

```
t_stramlijn(X1,Y1,Z1,X2,Y2,Z2):-
trans(X1,Y1,Z1,SS1,ST1),
trans(X2,Y2,Z2,SS2,ST2),
wgraph(ruimtelijk_ontwerp,pen_color(255,0,0)),
wgraph(ruimtelijk_ontwerp,polyline(SS1-ST1,SS2-ST2)),
SS1q is SS1+10,
wgraph(ruimtelijk_ontwerp,textout(SS1q,ST1,'(')),
SS1b is SS1q+10,
SS1c is SS1b+60,
SS1d is SS1c+10,
SS1e is SS1d+60,
SS1f is SS1e+10,
SS1g is SS1f+60,
number(X1,T1),
number(Y1,T2),
number(Z1,T3),
wgraph(ruimtelijk_ontwerp,textout(SS1b,ST1,T1)),
wgraph(ruimtelijk_ontwerp,textout(SS1c,ST1,',')),
wgraph(ruimtelijk_ontwerp,textout(SS1d,ST1,T2)),
wgraph(ruimtelijk_ontwerp,textout(SS1e,ST1,',')),
wgraph(ruimtelijk_ontwerp,textout(SS1f,ST1,T3)),
wgraph(ruimtelijk_ontwerp,textout(SS1g,ST1,')')),
SS2q is SS2+10,
wgraph(ruimtelijk_ontwerp,textout(SS2q,ST2,'(')),
SS2b is SS2q+10,
SS2c is SS2b+60,
SS2d is SS2c+10,
SS2e is SS2d+60,
SS2f is SS2e+10,
SS2g is SS2f+60,
number(X2,TT1),
number(Y2,TT2),
number(Z2,TT3),
wgraph(ruimtelijk_ontwerp,textout(SS2b,ST2,TT1)),
wgraph(ruimtelijk_ontwerp,textout(SS2c,ST2,',')),
wgraph(ruimtelijk_ontwerp,textout(SS2d,ST2,TT2)),
wgraph(ruimtelijk_ontwerp,textout(SS2e,ST2,',')),
wgraph(ruimtelijk_ontwerp,textout(SS2f,ST2,TT3)),
wgraph(ruimtelijk_ontwerp,textout(SS2g,ST2,')')).
```

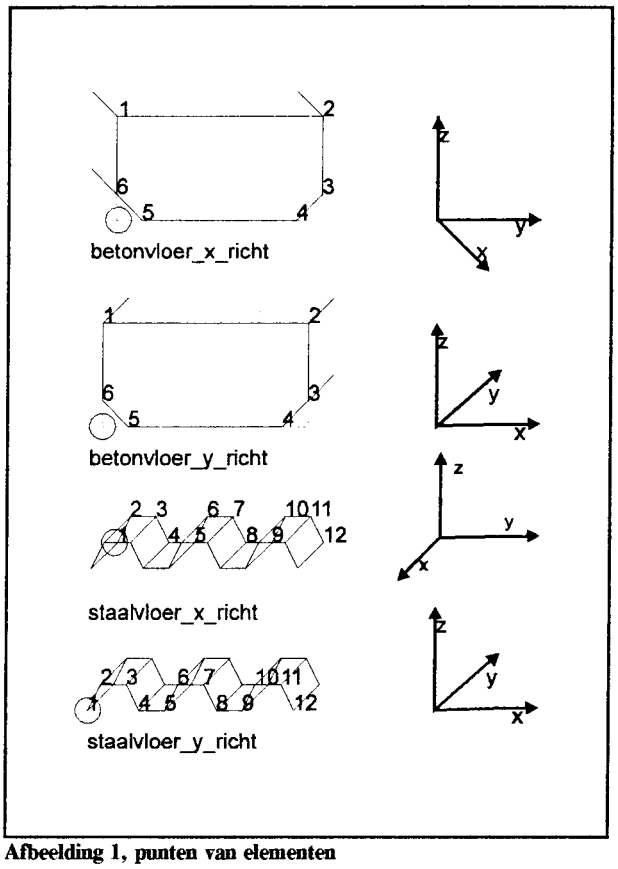
Bij t_stramlijn zorgen "textout"-predicaten ervoor dat er text op het scherm verschijnt. Met "number" worden getallen omgezet in karakters, zodat deze afgedrukt kunnen worden op het scherm.

```
t_betonvloer_x_richt(P,Q,R,Br):-
S1 is P, U1 is R+0.2, trans(S1,Q,U1,SS1,ST1), S1b is P+Br,
trans(S1b,Q,U1,SS1b,ST1b),
T2 is Q+1, U2 is U1, trans(S1,T2,U2,SS2,ST2), trans(S1b,T2,U2,SS2b,ST2b),
T3 is T2, U3 is R+0.1, trans(S1,T3,U3,SS3,ST3),trans(S1b,T3,U3,SS3b,ST3b),
```

```
T4 is T2-0.1, U4 is R, trans(S1,T4,U4,SS4,ST4),trans(S1b,T4,U4,SS4b,ST4b),
T5 is Q+0.1, trans(S1,T5,U4,SS5,ST5),trans(S1b,T5,U4,SS5b,ST5b),
trans(S1,Q,U3,SS6,ST6),trans(S1b,Q,U3,SS6b,ST6b),
wgraph(ko_ontwerp,pen_color(1,1,1)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS2-ST2,SS2b-ST2b,SS1b-ST1b,SS1-ST1)),
wgraph(ko_ontwerp,brush_color(150,150,150)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS1b-ST1b,SS6b-ST6b,SS6-ST6,SS1-ST1)),
wgraph(ko_ontwerp,poly-
gon(SS1b-ST1b,SS2b-ST2b,SS3b-ST3b,SS4b-ST4b,SS5b-ST5b,SS6b-ST6b,SS1b-ST1b)).

t_betonvloer_y_richt(P,Q,R,Br):-
U1 is R+0.2, trans(P,Q,U1,SS1,ST1), T1 is Q+Br, trans(P,T1,U1,SS1b,ST1b),
S2 is P+1, U2 is U1, trans(S2,Q,U2,SS2,ST2), trans(S2,T1,U2,SS2b,ST2b),
S3 is S2, U3 is R+0.1, trans(S3,Q,U3,SS3,ST3),trans(S3,T1,U3,SS3b,ST3b),
S4 is S2-0.1, U4 is R, trans(S4,Q,U4,SS4,ST4),trans(S4,T1,U4,SS4b,ST4b),
S5 is P+0.1, trans(S5,Q,U4,SS5,ST5),trans(S5,T1,U4,SS5b,ST5b),
trans(P,Q,U3,SS6,ST6),trans(P,T1,U3,SS6b,ST6b),
wgraph(ko_ontwerp,pen_color(1,1,1)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS2-ST2,SS2b-ST2b,SS1b-ST1b,SS1-ST1)),
wgraph(ko_ontwerp,brush_color(150,150,150)),
wgraph(ko_ontwerp,polygon(SS2-ST2,SS2b-ST2b,SS3b-ST3b,SS3-ST3,SS2-ST2)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS2-ST2,SS3-ST3,SS4-ST4,SS5-ST5,SS6-ST6,SS1-ST1)).

t_staalvloer_x_richt(P,Q,R,Br):-
P2 is P+Br,
trans(P,Q,R,S1,T1),
trans(P2,Q,R,S1b,T1b),
Y2 is Q+0.08, Z2 is R+0.2, trans(P,Y2,Z2,S2,T2), trans(P2,Y2,Z2,S2b,T2b),
Y3 is Y2+0.12, Z3 is Z2, trans(P,Y3,Z3,S3,T3),  trans(P2,Y3,Z3,S3b,T3b),
Y4 is Y3+0.08, Z4 is R, trans(P,Y4,Z4,S4,T4), trans(P2,Y4,Z4,S4b,T4b),
Y5 is Y4+0.12, Z5 is R, trans(P,Y5,Z5,S5,T5), trans(P2,Y5,Z5,S5b,T5b),
Y6 is Y5+0.08, Z6 is Z2, trans(P,Y6,Z6,S6,T6), trans(P2,Y6,Z6,S6b,T6b),
Y7 is Y6+0.12, Z7 is Z2, trans(P,Y7,Z7,S7,T7), trans(P2,Y7,Z7,S7b,T7b),
Y8 is Y7+0.08, Z8 is R, trans(P,Y8,Z8,S8,T8), trans(P2,Y8,Z8,S8b,T8b),
Y9 is Y8+0.12, Z9 is R, trans(P,Y9,Z9,S9,T9), trans(P2,Y9,Z9,S9b,T9b),
Y10 is Y9+0.08, Z10 is Z2, trans(P,Y10,Z10,S10,T10), trans(P2,Y10,Z10,S10b,T10b),
Y11 is Y10+0.12, Z11 is Z2, trans(P,Y11,Z11,S11,T11), trans(P2,Y11,Z11,S11b,T11b),
Y12 is Y11+0.08, Z12 is R, trans(P,Y12,Z12,S12,T12), trans(P2,Y12,Z12,S12b,T12b),
wgraph(ko_ontwerp,pen_color(1,1,1)),
wgraph(ko_ontwerp,polyline(S11b-T11b,S12b-T12b)),
wgraph(ko_ontwerp,brush_color(150,150,150)),
wgraph(ko_ontwerp,polygon(S10-T10,S11-T11,S11b-T11b,S10b-T10b,S10-T10)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,polygon(S10-T10,S9-T9,S9b-T9b,S10b-T10b,S10-T10)),
wgraph(ko_ontwerp,brush_color(255,255,255)),
wgraph(ko_ontwerp,polygon(S9-T9,S9b-T9b,S8b-T8b,S8-T8,S9-T9)),
wgraph(ko_ontwerp,polyline(S7b-T7b,S8b-T8b)),
wgraph(ko_ontwerp,brush_color(150,150,150)),
```

```
wgraph(ko_ontwerp,polygon(S6-T6,S7-T7,S7b-T7b,S6b-T6b,S6-T6)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,polygon(S6-T6,S5-T5,S5b-T5b,S6b-T6b,S6-T6)),
wgraph(ko_ontwerp,brush_color(255,255,255)),
wgraph(ko_ontwerp,polygon(S4-T4,S5-T5,S5b-T5b,S4b-T4b,S4-T4)),
wgraph(ko_ontwerp,polyline(S4b-T4b,S3b-T3b)),
wgraph(ko_ontwerp,brush_color(150,150,150)),
wgraph(ko_ontwerp,polygon(S2-T2,S3-T3,S3b-T3b,S2b-T2b,S2-T2)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,polygon(S1-T1,S2-T2,S2b-T2b,S1b-T1b,S1-T1)).

t_staalvloer_y_richt(P,Q,R,Br):-
QQ is Q+Br,
trans(P,Q,R,S1,T1),
trans(P,QQ,R,S1b,T1b),
X2 is P+0.08, Z2 is R+0.2, trans(X2,Q,Z2,S2,T2), trans(X2,QQ,Z2,S2b,T2b),
X3 is X2+0.12, Z3 is Z2, trans(X3,Q,Z3,S3,T3),  trans(X3,QQ,Z3,S3b,T3b),
X4 is X3+0.08, Z4 is R, trans(X4,Q,Z4,S4,T4), trans(X4,QQ,Z4,S4b,T4b),
X5 is X4+0.12, Z5 is R, trans(X5,Q,Z5,S5,T5), trans(X5,QQ,Z5,S5b,T5b),
X6 is X5+0.08, Z6 is Z2, trans(X6,Q,Z6,S6,T6), trans(X6,QQ,Z6,S6b,T6b),
X7 is X6+0.12, Z7 is Z2, trans(X7,Q,Z7,S7,T7), trans(X7,QQ,Z7,S7b,T7b),
X8 is X7+0.08, Z8 is R, trans(X8,Q,Z8,S8,T8), trans(X8,QQ,Z8,S8b,T8b),
X9 is X8+0.12, Z9 is R, trans(X9,Q,Z9,S9,T9), trans(X9,QQ,Z9,S9b,T9b),
X10 is X9+0.08, Z10 is Z2, trans(X10,Q,Z10,S10,T10), trans(X10,QQ,Z10,S10b,T10b),
X11 is X10+0.12, Z11 is Z2, trans(X11,Q,Z11,S11,T11), trans(X11,QQ,Z11,S11b,T11b),
X12 is X11+0.08, Z12 is R, trans(X12,Q,Z12,S12,T12), trans(X12,QQ,Z12,S12b,T12b),
wgraph(ko_ontwerp,pen_color(1,1,1)),
wgraph(ko_ontwerp,polyline(S8-T8,S10-T10)),
wgraph(ko_ontwerp,polyline(S9b-T9b,S10b-T10b)),
wgraph(ko_ontwerp,brush_color(150,150,150)),
wgraph(ko_ontwerp,polygon(S10-T10,S11-T11,S11b-T11b,S10b-T10b,S10-T10)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,polygon(S11-T11,S12-T12,S12b-T12b,S11b-T11b,S11-T11)),
wgraph(ko_ontwerp,brush_color(255,255,255)),
wgraph(ko_ontwerp,polygon(S9-T9,S9b-T9b,S8b-T8b,S8-T8,S9-T9)),
wgraph(ko_ontwerp,polyline(S5-T5,S6-T6)),
wgraph(ko_ontwerp,brush_color(150,150,150)),
wgraph(ko_ontwerp,polygon(S6-T6,S7-T7,S7b-T7b,S6b-T6b,S6-T6)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,polygon(S7-T7,S8-T8,S8b-T8b,S7b-T7b,S7-T7)),
wgraph(ko_ontwerp,brush_color(255,255,255)),
wgraph(ko_ontwerp,polygon(S4-T4,S5-T5,S5b-T5b,S4b-T4b,S4-T4)),
wgraph(ko_ontwerp,polyline(S1-T1,S2-T2)),
wgraph(ko_ontwerp,brush_color(150,150,150)),
wgraph(ko_ontwerp,polygon(S2-T2,S3-T3,S3b-T3b,S2b-T2b,S2-T2)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,polygon(S3-T3,S4-T4,S4b-T4b,S3b-T3b,S3-T3)).

t_staalkolom(VP,VQ,R,H,Kwadrant):-
((Kwadrant=1,P is VP, Q is VQ);
```

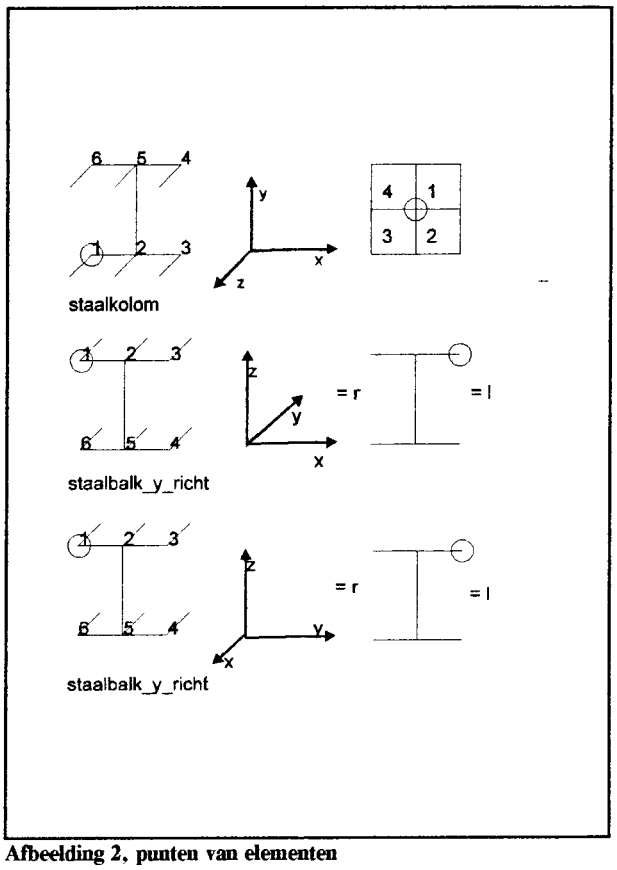
```
 (Kwadrant=2,P is VP, Q is VQ-0.2);
 (Kwadrant=3,P is VP-0.2, Q is VQ-0.2);
 (Kwadrant=4,P is VP-0.2, Q is VQ)),
RR is R+H,
trans(P,Q,R,SS1,ST1),trans(P,Q,RR,SS7,ST7),
S 2 i s
P+0.1,trans(S2,Q,R,SS2,ST2),trans(S2,Q,RR,S
S8,ST8),
S 3 i s
S2+0.1,trans(S3,Q,R,SS3,ST3),trans(S3,Q,RR,
SS9,ST9),
T 4 i s Q + 0 . 2 ,
trans(S3,T4,R,SS4,ST4),trans(S3,T4,RR,SS10,
ST10),
trans(S2,T4,R,SS5,ST5),trans(S2,T4,RR,SS11,
ST11),
trans(P,T4,R,SS6,ST6),trans(P,T4,RR,SS12,ST
12),
wgraph(kon_ontwerp,pen_color(0,0,0)),
wgraph(ko_ontwerp,brush_color(255,0,0)),
w g r a p h ( k o _ o n t w e r p , p o l y -
gon(SS4-ST4,SS6-ST6,SS12-ST12,SS10-ST10
,SS4-ST4)),
w g r a p h ( k o _ o n t w e r p , p o l y -
gon(SS2-ST2,SS5-ST5,SS11-ST11,SS8-ST8,S
S2-ST2)),
wgraph(ko_ontwerp,brush_color(255,0,0)),
w g r a p h ( k o _ o n t w e r p , p o l y -
gon(SS1-ST1,SS3-ST3,SS9-ST9,SS7-ST7,SS1-ST1)).
```



Afbeelding 1, punten van elementen

```
t_staalbalk_y_richt(PV,QV,R,L,Type):-
((Type=l, P is PV-0.2, Q is QV);
(Type=r, P is PV, Q is QV)),
QQ is Q+L,
trans(P,Q,R,SS1,ST1),trans(P,QQ,R,SS7,ST7),
S2 is P+0.1, trans(S2,Q,R,SS2,ST2),trans(S2,QQ,R,SS8,ST8),
S3 is S2+0.1,trans(S3,Q,R,SS3,ST3),trans(S3,QQ,R,SS9,ST9),
U4 is R-0.2,trans(S3,Q,U4,SS4,ST4),trans(S3,QQ,U4,SS10,ST10),
trans(S2,Q,U4,SS5,ST5),trans(S2,QQ,U4,SS11,ST11),
trans(P,Q,U4,SS6,ST6),trans(P,QQ,U4,SS12,ST12),
wgraph(ko_ontwerp,pen_color(0,0,0)),
wgraph(ko_ontwerp,brush_color(255,0,0)),
wgraph(ko_ontwerp,polygon(SS4-ST4,SS10-ST10,SS12-ST12,SS6-ST6,SS4-ST4)),
wgraph(ko_ontwerp,polygon(SS2-ST2,SS8-ST8,SS11-ST11,SS5-ST5,SS2-ST2)),
wgraph(ko_ontwerp,brush_color(255,0,0)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS3-ST3,SS9-ST9,SS7-ST7,SS1-ST1)).

t_staalbalk_x_richt(PV,QV,R,L,Type):-
((Type=l, P is PV, Q is QV-0.2);
(Type=r, P is PV, Q is QV)),
```

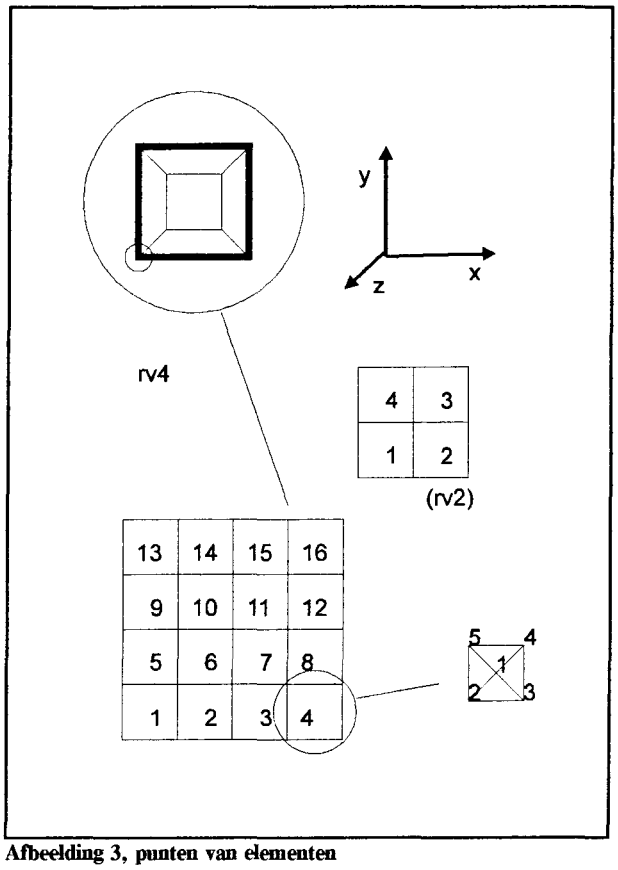
```
PP is P-L,
trans(P,Q,R,SS1,ST1),trans(PP,Q,R,SS7,ST7),
T2 is Q+0.1,
trans(P,T2,R,SS2,ST2),trans(PP,T2,R,SS8,ST8),
T3 is T2+0.1,trans(P,T3,R,SS3,ST3),trans(PP,T3,R,SS9,ST9),
U4 is R-0.2,trans(P,T3,U4,SS4,ST4),trans(PP,T3,U4,SS10,ST10),
trans(P,T2,U4,SS5,ST5),trans(PP,T2,U4,SS11,ST11),
trans(P,Q,U4,SS6,ST6),trans(PP,Q,U4,SS12,ST12),
wgraph(ko_ontwerp,pen_color(0,0,0)),
wgraph(ko_ontwerp,brush_color(255,0,0)),
wgraph(ko_ontwerp,polygon(SS4-ST4,SS10-ST10,SS12-ST12,SS6-ST6,SS4-ST4)),
wgraph(ko_ontwerp,polygon(SS2-ST2,SS8-ST8,SS11-ST11,SS5-ST5,SS2-ST2)),
wgraph(ko_ontwerp,brush_color(255,0,0)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS3-ST3,SS9-ST9,SS7-ST7,SS1-ST1)).
```



Afbeelding 2, punten van elementen

```
t_schijf_y_richt(VP,VQ,R,H,L,Type):-
((Type=r, P is VP, Q is VQ);
 (Type=l, P is VP-0.15,Q is VQ)),
QQ is Q+L,
trans(P,Q,R,SS1,ST1),trans(P,QQ,R,SS1b,ST1b),
S2 is P+0.15, trans(S2,Q,R,SS2,ST2),trans(S2,QQ,R,SS2b,ST2b),
U3 is R+H, trans(S2,Q,U3,SS3,ST3),trans(S2,QQ,U3,SS3b,ST3b),
trans(P,Q,U3,SS4,ST4),trans(P,QQ,U3,SS4b,ST4b),
wgraph(ko_ontwerp,pen_color(0,0,0)),
wgraph(ko_ontwerp,brush_color(255,175,255)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS2-ST2,SS3-ST3,SS4-ST4,SS1-ST1)),
wgraph(ko_ontwerp,polygon(SS2-ST2,SS3-ST3,SS3b-ST3b,SS2b-ST2b,SS2-ST2)),
wgraph(ko_ontwerp,polygon(SS3-ST3,SS4-ST4,SS4b-ST4b,SS3b-ST3b,SS3-ST3)).

t_schijf_x_richt(VP,VQ,R,H,L,Type):-
((Type=r, P is VP, Q is VQ);
 (Type=l, P is VP,Q is VQ-0.15)),
PP is P-L,
trans(P,Q,R,SS1,ST1),trans(PP,Q,R,SS1b,ST1b),
T2 is Q+0.15, trans(P,T2,R,SS2,ST2),trans(PP,T2,R,SS2b,ST2b),
U3 is R+H, trans(P,T2,U3,SS3,ST3),trans(PP,T2,U3,SS3b,ST3b),
trans(P,Q,U3,SS4,ST4),trans(PP,Q,U3,SS4b,ST4b),
```

```
wgraph(ko_ontwerp,pen_color(0,0,0)),
wgraph(ko_ontwerp,brush_color(255,175,255)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS2-ST2,SS3-ST3,SS4-ST4,SS1-ST1)),
wgraph(ko_ontwerp,polygon(SS4-ST4,SS3-ST3,SS3b-ST3b,SS4b-ST4b,SS4-ST4)),
wgraph(ko_ontwerp,polygon(SS1-ST1,SS4-ST4,SS4b-ST4b,SS1b-ST1b,SS1-ST1)).

t_betonkolom(VP,VQ,R,H,Kwadrant):-
((Kwadrant=1,P is VP, Q is VQ);
 (Kwadrant=2,P is VP, Q is VQ-0.2);
 (Kwadrant=3,P is VP-0.2,Q is VQ-0.2);
 (Kwadrant=4,P is VP-0.2, Q is VQ)),
RR is R+H,
trans(P,Q,R,SS1,ST1),trans(P,Q,RR,SS1b,ST1
b),
S2 is P+0.2,T3 is Q+0.2,
trans(S2,Q,R,SS2,ST2),trans(S2,Q,RR,SS2b,S
T2b),
trans(S2,T3,R,SS3,ST3),trans(S2,T3,RR,SS3b,
ST3b),
trans(P,T3,R,SS4,ST4),trans(P,T3,RR,SS4b,ST
4b),
wgraph(ko_ontwerp,pen_color(0,0,0)),
wgraph(ko_ontwerp,brush_color(200,200,200)),
wgraph(ko_ontwerp,poly-
gon(SS1-ST1,SS2-ST2,SS2b-ST2b,SS1b-ST1b
,SS1-ST1)),
wgraph(ko_ontwerp,poly-
gon(SS2-ST2,SS3-ST3,SS3b-ST3b,SS2b-ST2b
,SS2-ST2)),
wgraph(ko_ontwerp,poly-
gon(SS1b-ST1b,SS2b-ST2b,SS3b-ST3b,SS4b-
ST4b,SS1b-ST1b)).
```

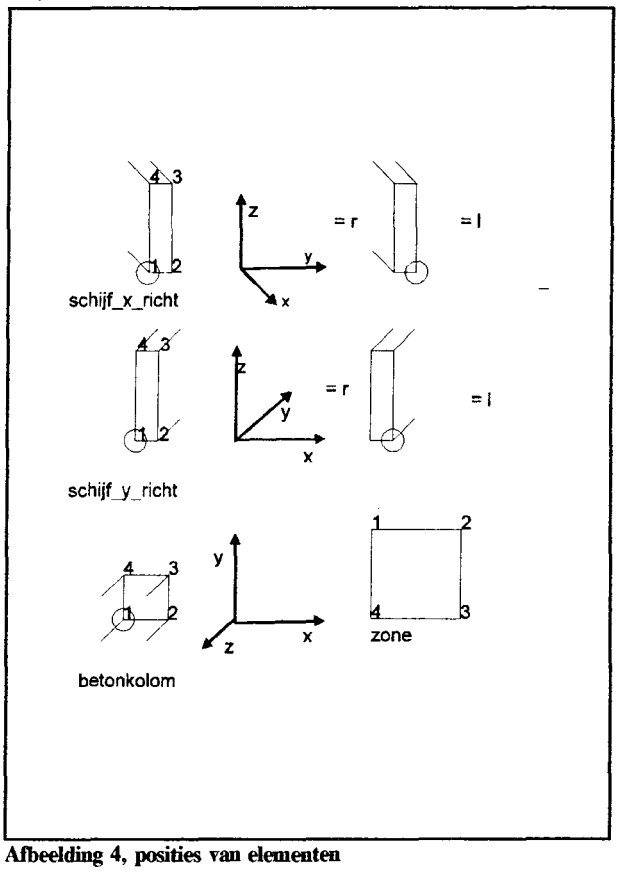


Afbeelding 3, punten van elementen

```
rve(P,Q,R):-
S2 is P-0.5, T2 is Q-0.5, trans(S2,T2,R,SS2,ST2),
S3 is S2+1, trans(S3,T2,R,SS3,ST3),
T4 is T2+1, trans(S3,T4,R,SS4,ST4),
trans(S2,T4,R,SS5,ST5),
RR is R-0.5,
trans(P,Q,RR,SS1,ST1),
wgraph(ko_ontwerp,pen_color(0,0,150)),
wgraph(ko_ontwerp,pen_width(2)),
wgraph(ko_ontwerp,polyline(SS2-ST2,SS3-ST3,SS4-ST4,SS5-ST5,SS2-ST2)),
wgraph(ko_ontwerp,polyline(SS2-ST2,SS1-ST1,SS4-ST4,SS5-ST5,SS1-ST1,SS3-ST3)),
wgraph(ko_ontwerp,pen_width(1)).

t_rv4(P,Q,R):-
S1 is P+0.5, S2 is S1+1, S3 is S2+1, S4 is S3+1,
T1 is Q+0.5, T2 is T1+1, T3 is T2+1, T4 is T3+1,
rve(S1,T1,R),rve(S1,T2,R),rve(S1,T3,R),rve(S1,T4,R),
```

```
rve(S2,T1,R),rve(S2,T2,R),rve(S2,T3,R),rve(S2,T4,R),
rve(S3,T1,R),rve(S3,T2,R),rve(S3,T3,R),rve(S3,T4,R),
rve(S4,T1,R),rve(S4,T2,R),rve(S4,T3,R),rve(S4,T4,R).

t_rv2(P,Q,R):-
S1 is P+0.5, S2 is S1+1,
T1 is Q+0.5, T2 is T1+1,
rve(S1,T1,R),rve(S1,T2,R),
rve(S2,T1,R),rve(S2,T2,R).
```

De twee predicaten "t_rv2" en "t_rv4" roepen de tekenroutine "rve" aan die kleine delen van het ruimtevakwerk (1*1 meter) tekent. Verder werken al deze predicaten op dezelfde manier. Een aantal omtrekpunten van het te tekenen voorwerp worden berekend met enkele rekenkundige functies. Deze punten, opeenvolgend genummerd, zijn ook te zien in de vier afbeeldingen 1, 2, 3 en 4, punten van elementen. Bij deze elementen is een cirkel te zien van het punt waaraan de positie van een element gerelateerd wordt. Heeft een element een bepaalde positie, dan is dat de positie van het omcirkeld punt.



Afbeelding 4, posities van elementen

```
trans(P,Q,R,SX,SY):-
VF is 40,
TFX is -600,
TFY is  700,
X is VF * P,
Y is VF * (-Q),
Z is VF * R,
SXF is (0.866025403*X - 0.5*Y)+TFX,
((integer(SXF), SX is SXF);(SX is fix(SXF))),
SYF is (0.5*X + 0.866025403*Y)+TFY-Z,
((integer(SYF), SY is SYF);(SY is fix(SYF))).
```

"trans" vertaalt drie real-world posities in twee schermposities. Eerst wordt de plattegrond getransleerd over een hoek van dertig graden met behulp van een transformatiemaxtrix. Vervolgens wordt de real-world-z-positie opgeteld bij de scherm-y-positie. De tekening wordt op het scherm verschoven door TFX en TFY en de VF (vermenigvuldigingsfactor) zorgt ervoor dat de tekening voldoende groot op het scherm komt.

```
teken_ruim([]).
```

```
teken_ruim(Ruim_ontwerp):-
pak2(Ruimte,Ruim_ontwerp,Ruim_ontwerp2),
pak2(A,Ruimte,B), pak2(B2,B,_), B2=[P,Q,R],
ruimte(A,Br,Hg,Laag),
laag(Laag,Dikte,Hoogte),
wgraph(ruimtelijk_ontwerp,pen_color(0,255,0)),
VF is 40, /* vermenigvuldiging voor x- en y-richting */
TFX is -600,
TFY is 700,
X is VF * P,
Y is VF * (-Q),
Z is VF * R,
Bra is VF * Br,
Hga is VF * Hg,
Dia is VF * Dikte,
X1 is (X-Bra/2),
Y1 is (Y+Hga/2),
X2 is (X+Bra/2),
Y2 is Y1, X3 is X2,
Y3 is (Y-Hga/2),
X4 is X1, Y4 is Y3,
X1T is fix((0.866025403*X1 - 0.5*Y1)+TFX),
Y1T is fix((0.5*X1 + 0.866025403*Y1)+TFY-Z),
X2T is fix((0.866025403*X2 - 0.5*Y2)+TFX),
Y2T is fix((0.5*X2 + 0.866025403*Y2)+TFY-Z),
X3T is fix((0.866025403*X3 - 0.5*Y3)+TFX),
Y3T is fix((0.5*X3 + 0.866025403*Y3)+TFY-Z),
X4T is fix((0.866025403*X4 - 0.5*Y4)+TFX),
Y4T is fix((0.5*X1 + 0.866025403*Y4)+TFY-Z),
Y1T3D is Y1T-Dia,
Y2T3D is Y2T-Dia,
Y3T3D is Y3T-Dia,
Y4T3D is Y4T-Dia,
wgraph(ruimtelijk_ontwerp,polyline(X3T-Y3T,X3T-Y3T3D)),
wgraph(ruimtelijk_ontwerp,polyline(X2T-Y2T,X2T-Y2T3D)),
wgraph(ruimtelijk_ontwerp,polyline(X1T-Y1T,X1T-Y1T3D)),
wgraph(ruimtelijk_ontwerp,polyline(X4T-Y4T,X4T-Y4T3D)),
wgraph(ruimtelijk_ontwerp,polyline(X1T-Y1T3D,X2T-Y2T3D,X3T-Y3T3D,X4T-Y4T3D,X1T-Y1T3D)),
wgraph(ruimtelijk_ontwerp,polyline(X1T-Y1T,X2T-Y2T,X3T-Y3T,X4T-Y4T,X1T-Y1T)),
teken_ruim(Ruim_ontwerp2),!.
```

"teken_ruim" tekent het ruimtelijk ontwerp in de window "ro_ontwerp". De afmetingen van de ruimte worden vermenigvuldigd met VF, waarna een ruimte getekend wordt die genomen is met pak2 uit de lijst "Ruim_ontwerp". In dit predicaat is het handige trans-predicaat nog niet verwerkt en zodoende worden een aantal translatieberekeningen uitgevoerd. Nadat enkele polylines getekend zijn wordt "teken_ruim" opnieuw aangeroepen om de volgende ruimte uit de lijst "Ruim_ontwerp" te tekenen en dit is de eerste ruimte van de lijst "Ruim_ontwerp2".

```
teken_tussenstand([]).

teken_tussenstand(Ruim_ontwerp):-
pak2(Ruimte,Ruim_ontwerp,Ruim_ontwerp2),
pak2(A,Ruimte,B), pak2(B2,B,_), B2=[P,Q,R],
ruimte(A,Br,Hg,Laag),
laag(Laag,Dikte,Hoogte),
wgraph(plattegrond,pen_color(0,255,0)),
VF is 40,
TFX is -400,
TFY is 1000,
X is VF * P,
Y is VF * (-Q),
Z is VF * R,
Bra is VF * Br,
Hga is VF * Hg,
XT is X+TFX,
YT is Y+TFY,
X1 is fix((X-Bra/2))+TFX,
Y1 is fix((Y+Hga/2))+TFY,
X2 is fix((X+Bra/2))+TFX,
Y2 is Y1, X3 is X2,
Y3 is fix((Y-Hga/2))+TFY,
X4 is X1, Y4 is Y3,
wgraph(plattegrond,polyline(X1-Y1,X2-Y2,X3-Y3,X4-Y4,X1-Y1)),
wgraph(plattegrond,textout(XT,YT,A)),
teken_tussenstand(Ruim_ontwerp2),!.
```

"teken_tussenstand" werkt hetzelfde als "teken_ruim" alleen wordt nu naar de window "plattegrond" een plattegrond van het ontwerp verzonden. Als 1 ruimte, gepakt door "pak2", getekend is wordt "teken_tussenstand" opnieuw aangeroepen om de volgende ruimte van de lijst te tekenen.

```
teken_tussenstandver(Ruim_ontwerp,N,NN):-
pak(Ruimte,Ruim_ontwerp,Ruim_ontwerp2),
pak2(A,Ruimte,B), pak2(B2,B,_), B2=[P,Q,R],
ruimte(A,Br,Hg,Laag),
Laag=NN,
wgraph(plattegrond,pen_color(0,255,0)),
laag(Laag,Dikte,Hoogte),
VF is 40,
TFX is -400,
TFY is 1000,
X is VF * P,
Y is VF * (-Q),
Z is VF * R,
Bra is VF * Br,
Hga is VF * Hg,
XT is X+TFX,
```

```
YT is Y+TFY,
X1 is fix((X-Bra/2))+TFX,
Y1 is fix((Y+Hga/2))+TFY,
X2 is fix((X+Bra/2))+TFX,
Y2 is Y1, X3 is X2,
Y3 is fix((Y-Hga/2))+TFY,
X4 is X1, Y4 is Y3,
wgraph(plattegrond,polyline(X1-Y1,X2-Y2,X3-Y3,X4-Y4,X1-Y1)),
teken_tussenstandver(Ruim_ontwerp2,N,NN).

teken_tussenstandver(Ruim_ontwerp,N,NN):-
pak(Ruimte,Ruim_ontwerp,Ruim_ontwerp2),
pak2(A,Ruimte,B), pak2(B2,B,_), B2=[P,Q,R],
ruimte(A,Br,Hg,Laag),
Laag=N,
wgraph(plattegrond,pen_color(255,0,0)),
laag(Laag,Dikte,Hoogte),
VF is 40,
TFX is -400,
TFY is 1000,
X is VF * P,
Y is VF * (-Q),
Z is VF * R,
Bra is VF * Br,
Hga is VF * Hg,
XT is X+TFX,
YT is Y+TFY,
X1 is fix((X-Bra/2))+TFX,
Y1 is fix((Y+Hga/2))+TFY,
X2 is fix((X+Bra/2))+TFX,
Y2 is Y1, X3 is X2,
Y3 is fix((Y-Hga/2))+TFY,
X4 is X1, Y4 is Y3,
wgraph(plattegrond,polyline(X1-Y1,X2-Y2,X3-Y3,X4-Y4,X1-Y1)),
wgraph(plattegrond,textout(XT,YT,A)),
teken_tussenstandver(Ruim_ontwerp2,N,NN).

teken_tussenstandver(Ruim_ontwerp2,N,NN).
```

"teken_tussenstandver" kent 3 predicaten. De tweede variabele van het predicaat geeft het lagennummer aan waarvan de plattegrond getekend moet worden. De derde variabele geeft het lagennummer aan van de laag onder te in beginsel te tekenen laag. Deze laag zal met een andere kleur getekend worden. Telkens wordt met "pak2" een ruimte genomen uit de lijst. Nu wordt opnieuw "teken_tussenstandver" aangeroepen en gekeken of deze laag op N of NN ligt. Is dit het geval dan wordt de ruimte getekend. Is dit niet het geval dan wordt het feit "teken_tussenstandver" waar. Als de lijst geheel afgewerkt is en dus leeg, zal "teken_tussenstand" waar worden door het eerste feit "teken_tussenstand".

```
punt_op_schijf(Lijst_elementen,Punt):-
Punt = [X,Y,Z],
pak(Element,Lijst_elementen,Nlijst),
Element = [schijf,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],
Z = Z3,
((X = X1, Y >= Y1, Y =< Y2);(Y = Y1, X >= X1, X =< X2)).
```

Het predicaat "punt_op_schijf" bekijkt of een punt op een schijf uit de elementenlijst van "eindontwerp" ligt. Het punt bestaat uit een x-, y- en z-coordinaat en er wordt een element uit de elementenlijst genomen met "pak". Vervolgens wordt gekeken of de z-coordinaten van het punt en de bovenrand van de schijf identiek zijn. De schijf is opgebouwd uit 4 punten met wederom 3 coordinaten. De 4 punten zijn als volgt gevormd. Begonnen wordt met, indien de schijf in x-richting ligt, het punt met de laagste z-coordinaat en laagste x-coordinaat, waarna het tweede punt het punt is met dezelfde y- en z-waarden maar met hogere x-waarde. Het derde punt is het tweede punt met de hogere z-waarde, waarna het vierde punt boven het eerste punt ligt. Begonnen wordt met, indien de schijf in y-richting ligt, het punt met de laagste z-coordinaat en laagste y-coordinaat, waarna het tweede punt het punt is met dezelfde x- en z-waarden maar met hogere y-waarde. Het derde punt is het tweede punt met de hogere z-waarde, waarna het vierde punt boven het eerste punt ligt. In de clause "punt_op_schijf" wordt vervolgens met een arithmische vergelijking bekeken of het punt op de schijf ligt.

```
twee_punten_op_schijf(Lijst_elementen,Punt1,Punt2):-
Punt1 = [X,Y,Z],
Punt2 = [P,Q,R],
pak(Element,Lijst_elementen,Nlijst),
Element = [schijf,X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],
Z = Z3,
R = Z3,
((X = X1, Y >= Y1, Y =< Y2);(Y = Y1, X >= X1, X =< X2)),
((P = X1, Q >= Y1, Q =< Y2);(P = Y1, Q >= X1, Q =< X2)).
```

Het predicaat "twee_punten_op_schijf" bekijkt of twee punten op dezelfde schijf uit de elementenlijst van "eindontwerp" liggen. Dit predicaat is wat betreft de werking gelijk aan "punt_op_schijf".

```
punt_op_kolom(Lijst_elementen,Punt):-
Punt = [X,Y,Z],
pak(Element,Lijst_elementen,Nlijst),
Element = [k,X1,Y1,Z1,X2,Y2,Z2],
X = X2,
Y = Y2,
Z = Z2.
```

"punt_op_kolom" kijkt of een punt bovenop een kolom ligt. Een kolom wordt gerepresenteerd door twee punten met x-, y- en z-waarden. Het eerste punt is altijd het punt met de laagste z-waarde.

```
ruimte_binnen_zone(Ruimte,Positie,Lijst_zones):-
Positie = [X,Y,Z],
ruimte(Ruimte,Br,Hg,Laag),
laag(Laag,Dikte,Hoogte),
pak(Zone,Lijst_zones,Nlijst),
pak2(Zonegetallen,Zone,Nlijst2),
Zonegetallen = [X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],
R is Z1+Hoogte,
Z = R,
K is fix(X - 0.5*Br), K >= X1,
L is fix(X + 0.5*Br), L =< X2,
M is fix(Y + 0.5*Hg), M =< Y1,
N is fix(Y - 0.5*Hg), N >= Y3,!.
```

"ruimte_binnen_zone" kijkt of een ruimte met een bepaalde positie binnen een van de zones ligt genoemd in de zonelijst. Een zone wordt gerepresenteerd door vier punten met x-, y-, en z-waarden. De z-waarden van de vier punten zullen altijd gelijk aan elkaar zijn. Een zone heeft als z-waarde de onderkant van een laag. De zones in de zonelijst kunnen over meerdere lagen verspreid zitten (een gedachte voor de toekomst). De dikte van een ruimtelaag wordt opgeteld bij de hoogte van de zone en zo wordt de z-waarde die een positie van een ruimte behoort te hebben gevonden om binnen de zone te vallen. Merk op dat zo zones van andere lagen uitgesloten worden van medewerking. Omdat "pak" een willekeurig element kan nemen uit een lijst, zullen zo nodig alle zones onderzocht worden door backtracking. Op het eind wordt door een expressie onderzocht of de ruimte binnen de zone valt. Konstruktief is deze voorwaarde niet noodzakelijk en kan zelfs belemmerend werken. Deze voorwaarde is nu slechts ingebouwd ter beperking van het aantal mogelijkheden.

```
ko_lagen(N):-
bagof(P,ruimte3(P,N),Laaglijst),
ontwerp3(Lijstro,Lijstzones,Lijstelementen),
not(marker(N)),
plaats_ruimten(N,Lijstro,Lijstzones,Lijstelementen,Laaglijst,[]),
laag2(Positielijst),
append(Positielijst,Lijstro,Nlijstro),
interactief3(N,Nlijstro,Lijstzones,Lijstelementen),
retract(ontwerp3(Lijstro,Lijstzones,Lijstelementen)),
schoon,
fail.

ko_lagen(N):-
retractall(ontwerp3(A,B,C)),
schoon.
```

"ko_lagen(N)" is een belangrijk predicaat. N geeft de laag aan waar de ruimten geplaatst moeten worden. Eerst worden alle ruimten die op de laag N geplaatst moeten worden verzameld door "bagof" en "ruimte3" in de lijst Laaglijst. Een bestaand ruimtelijk ontwerp wordt aangeroepen door "eindontwerp". Nu moeten de ruimten geplaatst worden met "plaats_ruimten". Aan de ruimtelijke voorwaarden en de eis dat ruimten elkaar niet mogen overlappen wordt voldaan door de predicaten "ro_niet_overlap" en "ro_rvw". We gebruiken fail om te zorgen dat alle mogelijkheden onderzocht worden. Zijn alle mogelijkheden onderzocht dan wordt ko_lagen(_) toch waar omdat dit als feit is ingevoerd.

```
plaats_ruimten(_,_,_,_,[],Positielijst):-
assert(laag2(Positielijst)).

plaats_ruimten(N,Lijstro,Lijstzones,
Lijstelementen,Laaglijst,Positielijst):-
pak2(Ruimte,Laaglijst,Nlijst),
onderzoek(Type,Ruimte),
plaats(Ruimte,Positie,Lijstelementen,N),
ruimte_binnen_zone(Ruimte,Positie,Lijstzones),
ko_rvw(Ruimte,Positie,Lijstelementen,Type),
ro_niet_overlap(Ruimte,Positie,Positielijst),
ro_rvw(Ruimte,Positie,Positielijst),
voegbij([Ruimte,Positie],Positielijst,Npositielijst),
plaats_ruimten(N,Lijstro,Lijstzones,Lijstelementen,Nlijst,Npositielijst),
not(marker(N)).

plaats_ruimten(A,B,C,D,E):-
fail.
```

"plaats_ruimten" heeft als eerste variabele het ruimtelagennummer. De tweede, derde en vierde variabele zijn bekend. De lijst Laaglijst bestaat uit de te plaatsen ruimten op de laag. De lijst "Positielijst" wordt gebruikt om de posities van de geplaatste ruimten in op te slaan. Eerst wordt een ruimte met "pak" genomen en wordt deze ruimte onderzocht met "onderzoek": nu is bekend of de ruimte een vierkant is of een rechthoek. Deze informatie geeft de variabele Type. De ruimte wordt een plaats gegeven met "plaats" hetgeen wil zeggen dat de linkerbovenhoek van de ruimte op een kolom of op een schijf neergezet wordt. Vandaar dat ook de lijst Lijstelementen meegenomen wordt met "plaats". Er wordt bekeken of de ruimte met de plaats binnen een zone valt met "ruimte_binnen_zone" en vervolgens wordt bekeken of aan de mogelijkheden tot plaatsing (zoals deze te zien zijn in de afbeelding" wordt voldaan met "ko_rvw". Is dit alles het geval dan kan de positie toegevoegd aan de lijst Positielijst worden en wordt zo de lijst "Npositielijst" gevormd. Nu kan doorgegaan worden met het plaatsen van de overige ruimten. Als de lijst met te plaatsen ruimten "Laaglijst" leeg is dan wordt ervoor gezorgd dat de positielijst wordt opgeslagen met "assert" zoals te zien is in de eerste clause van "plaats_ruimten".

```
onderzoek(Type,Ruimte):-
ruimte(Ruimte,Br,Hg,Laag),
```

```
((Hg = Br, Type is 1); (Hg =\= Br, Type is 2)),!.
```

"onderzoek" bekijkt of een ruimte vierkant of rechthoekig is. Als de hoogte van de ruimte gelijk is aan de breedte is de ruimte vierkant. In alle andere gevallen is de ruimte rechthoekig. In dit geval noemen we vierkant "1" en rechthoekig "2".

```
plaats(Ruimte,Positie,Lijstelementen,Laag):-
laag(Laag,_,Hoogte),
ruimte(Ruimte,Br,Hg,Laag),
pak(Element,Lijstelementen,Nlijst),
pak2(Type,Element,Rest),
Type=schijf,
Rest=[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],
Y1=Y2, Z3=Hoogte,
T is X2-X1+1, count(N,T),
P is N+X1-1,
P2 is fix(P+0.5*Br),
Q2 is fix(Y1-0.5*Hg),
Positie = [P2,Q2,Z3].

plaats(Ruimte,Positie,Lijstelementen,Laag):-
laag(Laag,_,Hoogte),
ruimte(Ruimte,Br,Hg,Laag),
pak(Element,Lijstelementen,Nlijst),
pak2(Type,Element,Rest),
Type=schijf,
Rest=[X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3,X4,Y4,Z4],
Z3=Hoogte,
X1=X2,
T is Y2-Y1+1, count(N,T),
Q is N+Y1-1,
P2 is fix(X1+0.5*Br),
Q2 is fix(Q-0.5*Hg),
Positie = [P2,Q2,Z3].

plaats(Ruimte,Positie,Lijstelementen,Laag):-
laag(Laag,_,Hoogte),
ruimte(Ruimte,Br,Hg,Laag),
pak(Element,Lijstelementen,Nlijst),
pak2(Type,Element,Rest),
Type=k,
Rest=[X1,Y1,Z1,X2,Y2,Z2],
Z2=Hoogte,
X is fix(X2+0.5*Br),
Y is fix(Y2-0.5*Hg),
Positie = [X,Y,Z2].
```

Er kan op meerdere manieren een ruimte geplaatst worden door de linkerbovenhoek op een kolom te plaatsen of op een schijf te plaatsen. Bij het eerste predicaat "plaats" wordt

de posities van een kolom genomen. Om het middelpunt van een ruimte te kunnen vormen moet bij deze posities nog de halve breedte of halve hoogte van de ruimte verwerken worden. De predicaten "fix" ronden real getallen af tot integers.

Bij het tweede predicaat "plaats" wordt een ruimtehoek op een schijf geplaatst. Er wordt begonnen met met een schijf in y-richting (waarbij X1=X2) en daarna met het punt met de laagste y-waarde. Vervolgens worden met "count" bij de y-waarde waarden opgeteld totdat de grootste y-waarde op de schijf bereikt wordt. Omdat de standaard "count" bij 1 begint met tellen, wordt bij het telbereik van count 1 opgeteld en wordt van de som van laagste y-waarde en telbereik 1 afgehaald. De derde clause "plaats" werkt op dezelfde wijze als de tweede clause, alleen nu voor schijven in de x-richting (Y1=Y2).

```
ko_rvw(A,B,C,1):-ko_rvw2(A,B,C,1),!.

ko_rvw2(Ruimte,Positie,Lijstelementen,1):-
rb2(Ruimte,Positie,Punt),
(punt_op_kolom(Lijstelementen,Punt);
punt_op_schijf(Lijstelementen,Punt)),
ro2(Ruimte,Positie,Punt2),
(punt_op_kolom(Lijstelementen,Punt2);
punt_op_schijf(Lijstelementen,Punt2)),
lo2(Ruimte,Positie,Punt3),
(punt_op_kolom(Lijstelementen,Punt3);
punt_op_schijf(Lijstelementen,Punt3)).

ko_rvw(A,B,C,2):-ko_rvw2(A,B,C,2),!.

ko_rvw2(Ruimte,Positie,Lijstelementen,2):-
lb2(Ruimte,Positie,Punt1),
rb2(Ruimte,Positie,Punt2),
ro2(Ruimte,Positie,Punt3),
lo2(Ruimte,Positie,Punt4),
twee_punten_op_schijf(Punt1,Punt2,Lijstelementen),
punt_op_kolom(Punt3,Lijstelementen),
punt_op_kolom(Punt4,Lijstelementen).

ko_rvw2(Ruimte,Positie,Lijstelementen,2):-
lb2(Ruimte,Positie,Punt1),
rb2(Ruimte,Positie,Punt2),
ro2(Ruimte,Positie,Punt3),
lo2(Ruimte,Positie,Punt4),
twee_punten_op_schijf(Punt3,Punt4,Lijstelementen),
punt_op_kolom(Punt1,Lijstelementen),
punt_op_kolom(Punt2,Lijstelementen).

ko_rvw2(Ruimte,Positie,Lijstelementen,2):-
lb2(Ruimte,Positie,Punt1),
rb2(Ruimte,Positie,Punt2),
```

```
ro2(Ruimte,Positie,Punt3),
lo2(Ruimte,Positie,Punt4),
twee_punten_op_schijf(Punt1,Punt2,Lijstelementen),
twee_punten_op_schijf(Punt3,Punt4,Lijstelementen).

ko_rvw2(Ruimte,Positie,Lijstelementen,2):-
lb2(Ruimte,Positie,Punt1),
rb2(Ruimte,Positie,Punt2),
ro2(Ruimte,Positie,Punt3),
lo2(Ruimte,Positie,Punt4),
punt_op_kolom(Punt1,Lijstelementen),
punt_op_kolom(Punt2,Lijstelementen),
punt_op_kolom(Punt3,Lijstelementen),
punt_op_kolom(Punt4,Lijstelementen).
```

"ko_rvw" voor de vierkante ruimten (te zien aan Type=1) neemt het punt rechtsboven van de ruimte met "rb2" en kijkt of dit punt op een kolom ligt of dat dit punt op een schijf ligt met "punt_op_kolom" en "punt_op_schijf" en vervolgens wordt dit ook nagegaan bij de andere 2 hoekpunten.

"ko_rvw" voor de rechthoekige ruimten biedt meer mogelijkheden. Er kunnen twee punten op kolommen staan en twee punten op een schijf, maar nu wel op dezelfde schijf. Let wel, dit is nodig omdat de gehele schijf de ruimterand moet ondersteunen!. De andere "ko_rvw"-clauses zijn evenals deze beschreven "ko_rvw"-clause goed te begrijpen in samenhang met de clause "ko_rvw" voor vierkante ruimten.

```
lb2(Ruimte,[X,Y,Z],[P,Q,R]):-
ruimte(Ruimte,Br,Hg,Laag),
R is Z,
Q is fix(Y+0.5*Hg),
P is fix(X-0.5*Br).

rb2(Ruimte,[X,Y,Z],[P,Q,R]):-
ruimte(Ruimte,Br,Hg,Laag),
R is Z,
Q is fix(Y+0.5*Hg),
P is fix(X+0.5*Br).

ro2(Ruimte,[X,Y,Z],[P,Q,R]):-
ruimte(Ruimte,Br,Hg,Laag),
R is Z,
Q is fix(Y-0.5*Hg),
P is fix(X+0.5*Br).

lo2(Ruimte,[X,Y,Z],[P,Q,R]):-
ruimte(Ruimte,Br,Hg,Laag),
R is Z,
Q is fix(Y-0.5*Hg),
```

```
P is fix(X-0.5*Br).
```

"lb2", "rb2", "ro2" en "lo2" zijn identiek aan de eerder besproken clauses "lb", "rb", "ro" en "lo" met als verandering dat de eerst genoemde clauses resultaten leveren bij een eveneens gegeven punt. De laatst genoemde clauses hadden de database nodig om punten te kunnen retourneren.

```
interactief3(N,Nlijstro,Lijstzones,Lijstelementen):-
wcreategraphics(plattegrond,255,255,255),
NN is N-1,
teken_tussenstandver(Nlijstro,N,NN),
nl,
nl,write('De groene ruimten liggen op de laag onder de nu te ontwerpen laag,'),
nl,write('de rode ruimten zijn nu geplaatst.'),
nl,
nl,write('Ontwerp niet meenemen naar volgende ontwerpstap: a.'),
nl,write('Ontwerp meenemen naar volgende ontwerpstap: b.'),
nl,write('Ontwerp aanpassen (rood aangegeven ruimten) en daarna meenemen naar volgende
ontwerpstap: c.'),
nl,write('Alle nu nog komende ontwerpen inclusief deze overslaan en door naar volgende ontwerp-
stap: d.'),
nl,nl, write('Uw keuze: '),
read(A),!,
((A=d, assert(marker(N)));
 (A=b, assert(ontwerp(Nlijstro,Lijstzones,Lijstelementen)));
 (A=a);
 (A=c, interactief4(Nlijstro,Lijst2,N), assert(ontwerp(Lijst2,Lijstzones,Lijstelementen)))),
wdeletegraphics(plattegrond),!.

interactief4(Lijst,Lijst2,N):-
etaleer(Lijst),
nl,
nl,write('Het wijzigen van een ontwerp kan op grond van flexibiliteit zonder'),
nl,write('rekening te houden met de voorwaarden bij de invoer. Let goed op dat ruimten'),
nl,write('elkaar bijvoorbeeld kunnen overlappen of los van elkaar kunnen liggen.'),
nl,
nl,write('Neem alleen ruimten op de huidige laag.'),
nl,
nl,write('Welke ruimte wilt U wijzigen van plaats, geef alleen de naam van de ruimte'),
nl,write('gevolgd door een punt en een return: '),
read(A),
pak([A,[P,Q,R]],Lijst,Tussenlijst),
nl,
nl,write('Geef de x-waarde van de nieuwe positie van ruimte '), write(A), write(': '),
read(B),
nl,write('Geef de y-waarde van de nieuwe positie: '),
read(C),
voegbij([A,[B,C,R]],Tussenlijst,Lijst2),
wdeletegraphics(plattegrond),
```

```
wcreategraphics(plattegrond),
NN is N-1,
teken_tussenstandver(Lijst2,N,NN),
nl,nl,write('Gezien, toets willekeurig teken, punt en return: '),
read(QQQ).

schoon:-
retractall(laag2(_)).
```

"interactief3" en "interactief4" werken volstrekt identiek gelijk aan "interactief" en "interactief2".

```
ro_niet_overlap(_,_,[]).
ro_niet_overlap(Ruimte,Positie,Positielijst):-
pak2([Ruimte2,Positie2],Positielijst,Nlijst),
voorwaarde(no,Ruimte,Positie,Ruimte2,Positie2),
ro_niet_overlap(Ruimte,Positie,Nlijst),!.
```

"ro_niet_overlap" toetst of een ruimte "Ruimte" niet de ruimten in de positielijst "Positielijst" overlapt, met behulp van "voorwaarde".

```
ro_rvw(_,_,[]).
ro_rvw(Ruimte,Positie,Positielijst):-
pak2([Ruimte2,Positie2],Positielijst,Nlijst),
((voorwaarde_univers(a,Ruimte,Ruimte2),
voorwaarde(a,Ruimte,Positie,Ruimte2,Positie2));
not(voorwaarde_univers(a,Ruimte,Ruimte2))),
ro_rvw(Ruimte,Positie,Nlijst),!.
```

"ro_rvw" neemt een ruimte uit de positielijst "Positielijst" en kijkt of een voorwaarde is van de "Ruimte" met deze genomen ruimte. Is dit het geval ("voorwaarde_univers") dan zullen deze ruimte ook aan deze voorwaarde moeten voldoen met "voorwaarde". Hierna, of indien er geen voorwaarde is, wordt de volgende ruimte uit de positielijst "Positielijst genomen".

```
gewichts([],A,A).

gewichtb([],A,A).

gewichts(Lijstelem,Staal,Oudgewicht):-
pak2(Elem,Lijstelem,Lijstelem2),
tels(Elem,Oudgewicht,Nieuwgewicht),
gewichts(Lijstelem2,Staal,Nieuwgewicht),!.

gewichtb(Lijstelem,Beton,Oudgewicht):-
pak2(Elem,Lijstelem,Lijstelem2),
telb(Elem,Oudgewicht,Nieuwgewicht),
gewichtb(Lijstelem2,Beton,Nieuwgewicht),!.
```

"gewichts" neemt een element van de lijst met elementen met "pak2" en zorgt door middel van het predicaat "tels" dat bij het "Oudgewicht" het gewicht van het element wordt bijgeteld zodat "Nieuwgewicht" ontstaat. "gewichtb" werkt precies gelijk aan "gewichts" alleen voor het materiaal beton.

```
telb(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
(Type=schijf_y;Type=schijf_x),
Elem2=[_,_,_,Hoogte,Lengte,_,Dischijf],
Gewicht is 2400 * Hoogte * Lengte * Dischijf,
Nieuwgewicht is Oudgewicht + Gewicht.

telb(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
(Type=schijf_y;Type=schijf_x),
Elem2=[_,_,_,Hoogte,Lengte,_,Dischijf],
Gewicht is 2400 * Hoogte * Lengte * Dischijf,
Nieuwgewicht is Oudgewicht + Gewicht.

telb(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
((Type=vloerbet_y);(Type=vloerbet_x)),
Elem2=[_,_,_,Br,Divloer],
Gewicht is 2400 * 1 * Br * Divloer,
Nieuwgewicht is Oudgewicht + Gewicht.

telb(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
Type=kolbet,
Elem2=[_,_,_,Hoogte,_,Breedte],
Gewicht is 2400 * Hoogte * Breedte * Breedte,
Nieuwgewicht is Oudgewicht + Gewicht.

telb(A,B,B).

tels(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
Type=rv2,
Elem2=[_,_,_,Vldikte],
Gewicht is 100* Vldikte * 2 * 2,
Nieuwgewicht is Oudgewicht + Gewicht.

tels(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
Type=rv4,
Elem2=[_,_,_,Vldikte],
Gewicht is 100* Vldikte * 4 * 4,
Nieuwgewicht is Oudgewicht + Gewicht.
```

```
tels(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
Type=kolstaal,
Elem2=[_,_,_,Hoogte,_,Dikol],
Gewicht is 200 * Hoogte * Dikol * Dikol,
Nieuwgewicht is Oudgewicht + Gewicht.

tels(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
((Type=balkstaal_x);(Type=balkstaal_y)),
Elem2=[_,_,_,Br,_,Dibalk],
Gewicht is 200 * Br * Dibalk * Dibalk,
Nieuwgewicht is Oudgewicht + Gewicht.

tels(Elem,Oudgewicht,Nieuwgewicht):-
pak2(Type,Elem,Elem2),
((Type=vloerstaal_y);(Type=vloerstaal_x)),
Elem2=[_,_,_,Breedte,Divloer],
Gewicht is 200 * 1 * Divloer * Breedte,
Nieuwgewicht is Oudgewicht + Gewicht.

tels(A,B,B).
```

Met gevalsonderscheiding zal 1 predicaat "tels" behandeld worden. Hiermee wordt ook "telb" als behandeld beschouwd. Met "pak2" wordt het type van het element genomen. In het laatste geval moet het type "vloerstaal-_y of _x" zijn. Het gewicht van deze plaat is nu 200 * 1 * de dikte van de vloer vermenigvuldigd met de breedte. Het nieuwe gewicht is het oude gewicht + het gewicht.