#6353

Een Intelligent object geörienteerd computer ontwerpsysteem voor werktuigbouwkundige installaties

*Ir. W. Zeiler**

An Intelligent object oriented Computer Aided Design system for HVAC systems

Inleiding
Bij het ontwerpen van installaties voor klimaatbeheersing in gebouwen is de invloed van de computer nog altijd vrij beperkt. Er wordt weliswaar met de computer getekend en gerekend, maar van gegevensintegratie met behulp van de computer is nog niet of nauwelijks sprake. Om de installaties te ontwerpen is zowel in regeltechnisch opzicht als in werktuigbouwkundig opzicht de nodige expertise vereist. Deze expertise bestaat niet alleen uit vuistregels maar ook uit kennis van de beschikbare componenten. Niet iedereen bezit deze kennis, zodat veel tijd tijdens het ontwerpen moet worden besteed aan het opzoeken van eigenschappen van componenten die in het systeem worden geplaatst. Als tijdens het ontwerp gebruik kan worden gemaakt van een geautomatiseerd systeem dat voorziet in deze behoefte aan specialistische kennis, kan hiermee een aanzienlijke verbetering van de doorlooptijd tijdens de engineering van een project worden bereikt.

Samenvatting
Dit artikel geeft een beschrijving van een intelligent CAD-systeem voor klimaatinstallaties.
Met behulp van een testsysteem, is bekeken of de opgezette structuur ervan voldoet. Het gegevensverwerkende gedeelte van het programma werd getest. Hierbij is gebleken dat de structuur naar tevredenheid functioneert en dat de functionaliteit van het systeem verder kan worden uitgebreid.
Het kernbegrip 'Object Oriented Programming', wordt nader toegelicht. Tevens is een verklarende woordenlijst toegevoegd voor de belangrijkste informatiebegrippen.

Summary
This article describes an intelligent CAD System for HVAC installations. A testing program has been used to evaluate the architecture of the system. Also the information processing part of the program was tested. It was found that the structure of the program is satisfactory and that functional performance can be extended.
The essentials of 'Object Oriented Programming' are explained. A glossary of terms, containing key expressions on data processing, is added.

Doelstelling

Door in het Intelligent Computer Aided Design (ICAD)-systeem kennis vast te leggen die voor het bouwen van een installatie nodig is, kan de gebruiker bij het ontwerpen worden bijgestaan met adviezen en controles.

De doelstelling is om de mogelijkheden met kennisregels in het ICAD-systeem op het gebied van meet- en regeltechniek te analyseren, te ontwerpen en in te voeren met een gestructureerde aanpak.
Hierbij is het belangrijk om ondersteuning aan de eindgebruiker te bieden tijdens de engineeringfase en een zo duidelijk mogelijk inzicht te verschaffen in de werking van, en de samenhang tussen, de componenten van de installatie. Deze ondersteuning vindt plaats op basis van advies tijdens het installatieontwerp en controle na het plaatsen van componenten in het installatieontwerp.

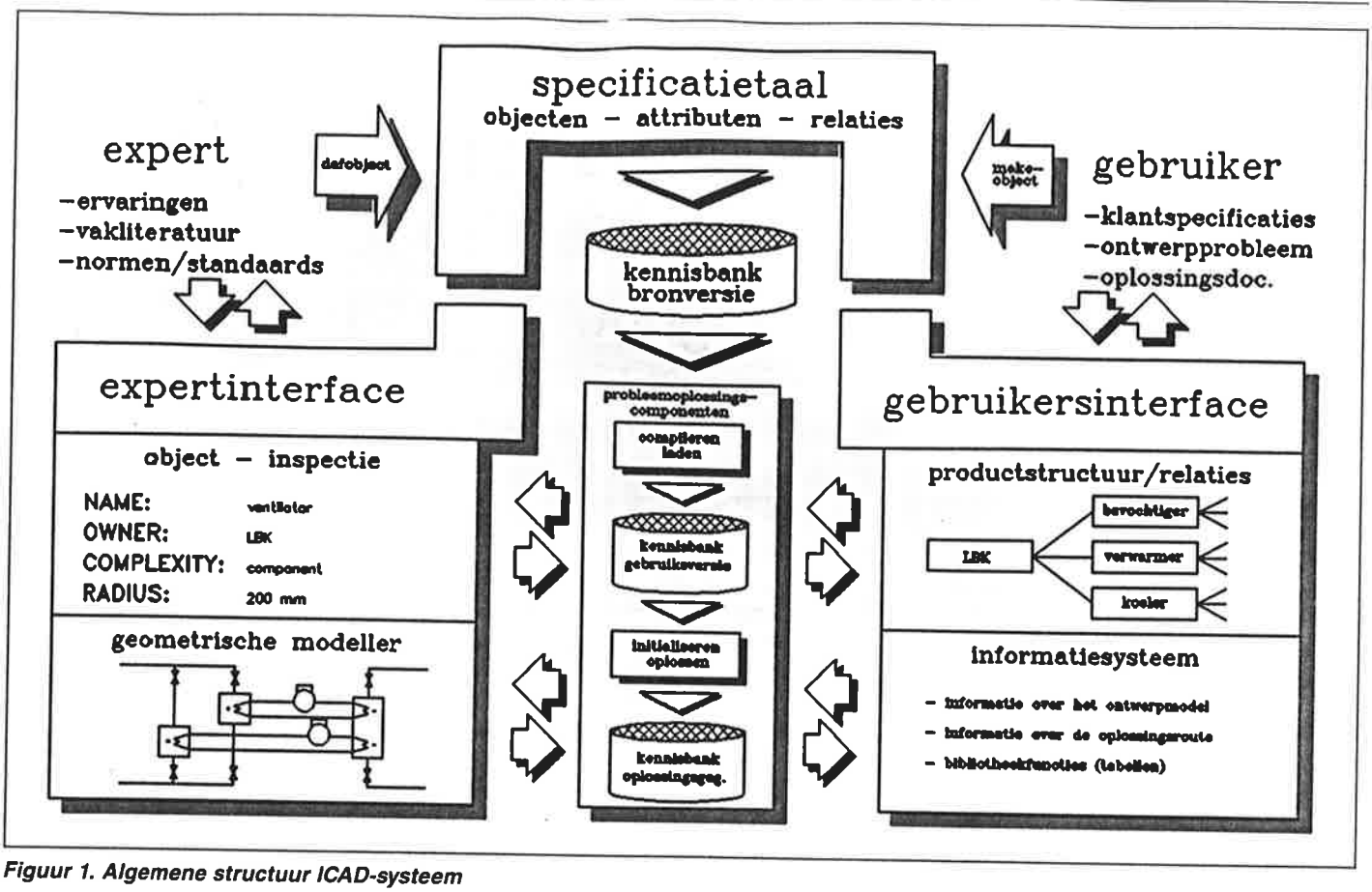
Globale oplossing

Het systeem is opgedeeld in een aantal onderdelen :
- Datastructuur;
- Graphical User Interface (GUI);
- Redeneermechanisme;
- Database Interface.

De totaal structuur van het systeem is te zien in fig. 1.
De GUI is onder te verdelen in een tweetal gedeelten : de gebruikersinterface en de expertinterface. In principe hoeft hier geen verschil tussen te zijn. Alleen voor het systeem heeft het gevolgen : de expert kan componenten en subsystemen definiëren, terwijl de eindgebruiker dit niet kan. Deze kan ze alleen opvragen en in het ontwerp plaatsen.
De datastructuur en het redeneermechanisme worden iets verder toegelicht.

Datastructuur
Gezien de grootte en de complexiteit van het ICAD project is het zeer belangrijk dat het flexibel wordt opgebouwd, zodat het in de loop van de tijd kan worden uitgebreid tot een steeds volwaardiger systeem.
Er is daarom voor een dynamische

* adjunct-directeur Kropman B.V. Rijswijk



Figuur 1. Algemene structuur ICAD-systeem

datastructuur gekozen.
Het ICAD-systeem moet verschillende, uiteenlopende taken gaan uitvoeren:
- Het selecteren van componenten, zoals pompen, afsluiters en regelafsluiters;
- Het uitwerken van het elektrisch hoofd- en stuurstroomontwerp, als het model van installatie bekend is;
- Het samenstellen van stuklijsten en begrotingen;
- Het presenteren van tussentijdse resultaten in de vorm van principeschema's;
- Het controleren van het ontwerp op verschillende aspecten (o.a. consistentie);
- Het documenteren van het verloop van het engineeringsproces.

De datastructuur van het ICAD-systeem moet zorgen voor voldoende ondersteuning en voor het waarborgen van de integriteit van de diverse gegevens.

Bij het ontwerpen van de datastructuur wordt met de volgende punten rekening gehouden:
- De installatie is hiërarchisch opgebouwd door middel van functionele abstractie;
- Iedere component in een te ontwerpen installatie moet éénduidig toegankelijk (adresseerbaar) zijn.
- Voor het doorrekenen van een systeem moeten er koppelingen kunnen worden aangebracht tussen de componenten;
- Iedere component heeft een grafische weergave;
- De datastructuur moet uniform zijn voor het gehele systeem.

Redeneermechanisme
Een redeneermechanisme is een mechanisme met behulp waarvan het systeem tot een oplossing van een probleem probeert te komen. In het geval van het ICAD-systeem betreft het probleem het bepalen van de samenstelling van een (sub)installatie. Daarnaast moeten alle waarden van de eigenschappen worden bepaald. Het redeneermechanisme zal aan de hand van aan het systeem toegevoegde kennis gaan proberen dit probleem op te lossen.

Een systeemcomponent bestaat uit een algemene omschrijving en de beschrijving van de eigenschappen van die component. Deze beschrijving kan alleen bestaan uit een waarde, maar het is ook mogelijk dat de waarde van de eigenschap afhankelijk is van andere eigenschappen. Om de waarde van de eigenschappen te kunnen bepalen is een 'parser' en 'interpreter' nodig. Een parser is een (deel van een) programma dat een aangeboden regel of formule op syntactische correctheid controleert en de semantiek ervan bepaalt. Een interpreter gebruikt de uitvoer van de parser en evalueert de regel of formule, met als resultaat de uitkomst van de regel of formule. Het is ook mogelijk om aan de hand van bepaalde gegevens uit de ontwerpomgeving van een component een bepaalde actie uit te voeren, zoals het kiezen van een component uit een bibliotheek. De samenstelling waaruit een (deel)systeem kan bestaan wordt weergegeven in een regel, deze regels zijn opgeslagen in de database. Tevens zijn er regels nodig voor de fysieke koppeling. Deze moeten vastleggen of een koppeling tussen twee componenten mag worden geplaatst. Ook deze regels worden vastgelegd in een database.

Database/Componenten-Bibliotheek
Alvorens de componenten en de kennis over de componenten en installaties in een database kunnen worden geplaatst, is het belangrijk te kijken naar hoe kennis kan worden gerepresenteerd. Het is hierbij ook van belang om de verschil-

lende soorten kennis op te splitsen in hierarchische niveaus. De representatie van de kennis is de eerste stap op weg naar een oplossing van het "probleem". Daarna zal met de vastgelegde kennis geredeneerd moeten gaan worden.

In geval van de object georiënteerde methode spitst het oplossen van het probleem zich toe op het invullen van de slotwaarden van de objecten. Hierbij is van belang welke weg er wordt gevolgd tijdens het invullen van de slots. Dit is zogenaamde strategische kennis : wat-wordt-wanneer-waarmee-gebruikt. Bij conventionele systemen wordt de loop van het oplossingsproces bepaald door van te voren geprogrammeerde controlestructuren. Als het een regel georiënteerd kennissysteem betreft dan zal een inferentiemechanisme bepalen hoe het proces verloopt. Een inferentiemechanisme is een mechanisme waarmee een oplossing voor een probleem kan worden gevonden. Een nadeel hiervan is dat het procesverloop niet geheel onafhankelijk is van de regelvolgorde van het systeem. Als er regels worden tussengevoegd kan het verloop veranderen, zodat het systeem opnieuw zal moeten worden getest. Bovendien zijn de verwerkingstijden bij complexere problemen vaak lang.

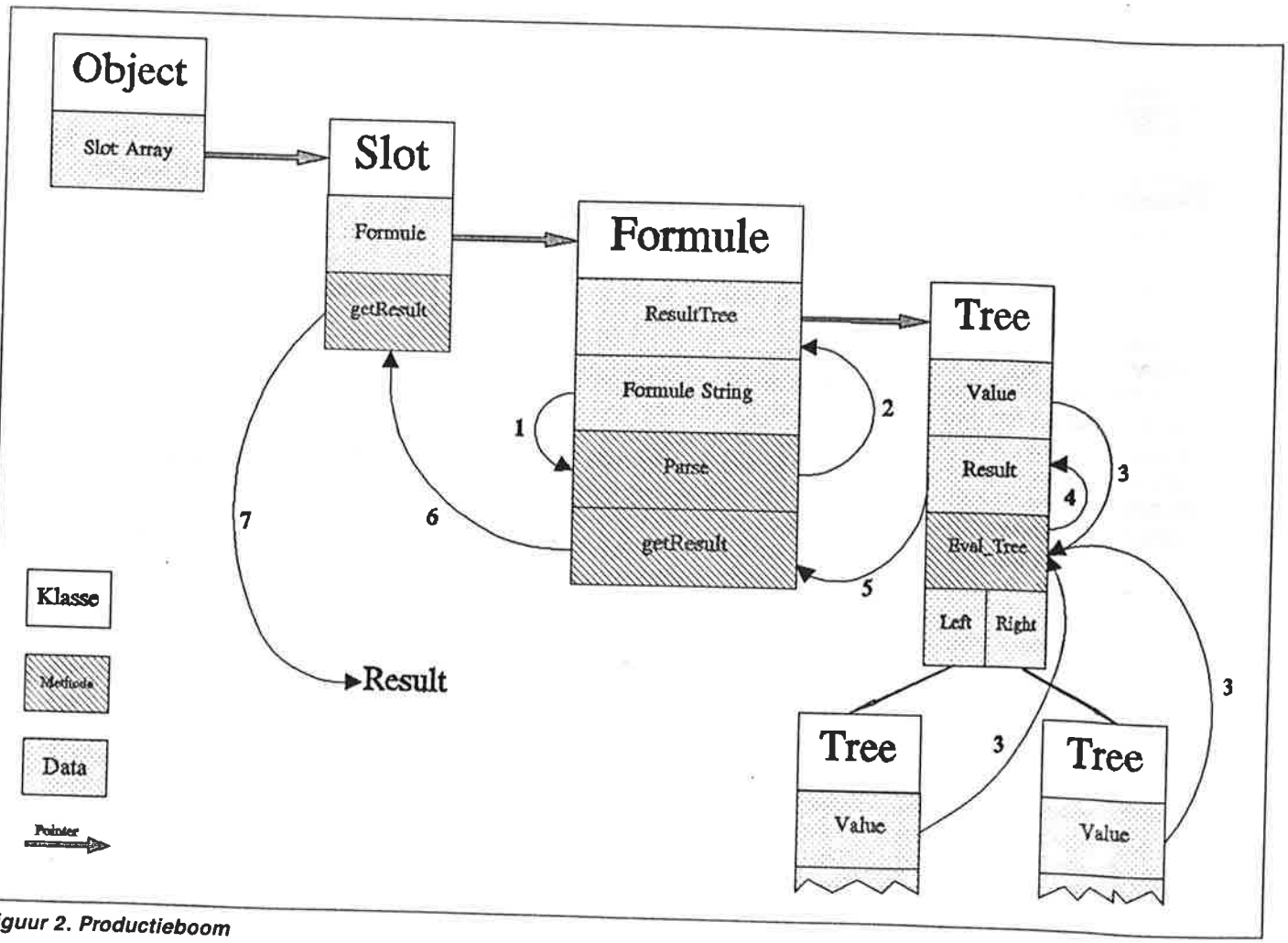
Kennisrepresentatie

Bij het ontwerpen van een installatie speelt de manipulatie van objecten constant een grote rol. Deze objecten kunnen gemanipuleerd worden met behulp van per object-soort vastliggend gedrag. Daarom is voor een object georiënteerde methode gekozen. De object georiënteerde methode kent immers ook de methoden die in het object ingekapseld zijn, samen met de data van dat object. Daarnaast is het ook een zeer complete methode, omdat het eigenlijk een combinatie is van de andere methoden.
Alle bovenbeschreven principes zijn in een object georiënteerde taal te implementeren, dit omdat aan de basis dezelfde objecten staan.
De objecten worden in een semantisch netwerk geplaatst.
De relaties in dit netwerk zijn als volgt :
- Has_Parts;
- Is_Part_Of;
- Is_Connected.

Deze relaties worden niet als data in het object opgenomen, maar in een database buiten het object.
De objecten zijn vastgelegd in zogenaamde 'atoms'. Voor iedere atom die wordt gebruikt, wordt als het ware een kopie gemaakt van een prototype. Dit prototype wordt gedefinieerd in de vorm van een zogenaamde 'klasse'. Een object van het type klasse wordt een 'instantiatie' van die klasse genoemd.

Een atom is nog geen object, maar een abstractie daarvan. Zo is het mogelijk dat een atom een hele verzameling van componenten representeert. Pas later zal een atom differentiëren tot een 'component' (meer of minder gedetailleerd). De eigenschappen van de atoms worden als data van het object beschouwd. Dat wil niet zeggen dat ze een vaststaande waarde hebben, maar dat het object zelf alleen maar deze waarden kan wijzigen.

Dit is ook het grootste verschil met de frame georiënteerde methode, waarbij eigenlijk iedereen waarden van een frame kan opvragen en wijzigen. Doordat de data in een object ingekapseld zijn, moeten deze worden benaderd via het



Figuur 2. Productieboom

object; er moet een boodschap naar het object worden verstuurd met een verzoek om de data een bepaalde waarde te geven, of om de waarde op te vragen. De slotwaarden worden bepaald door gebruik te maken van DDPS (Dependency Driven Problem Solving). Voor elke slotwaarde wordt de formule geparsed, dat wil zeggen worden de benodigde verdere verwijzingen afgelopen. Het resultaat hiervan is een productieboom. Deze boom heeft in de knooppunten operatoren of (attribuut)waarden. Aan de hand van deze productieboom kan de attribuutwaarde, die afhankelijk is van de formule, worden bepaald.
Als in één van de knooppunten van de boom een verwijzing naar een ander slot voorkomt, dan zal dit tot gevolg hebben dat de waarde van dit slot ook moet worden bepaald. Daarom heet de oplossingsmethode 'Dependancy Driven': aan de hand van de afhankelijkheden tussen slotwaarden worden alle slotwaarden uitgerekend.
Slotwaarden die niet afhankelijk zijn van andere attribuutwaarden en die niet bepalend zijn voor de waarde van een andere attribuut kunnen bij het problemsolving-proces buiten beschouwing worden gelaten. De waarde van deze attributen is kennelijk niet afhankelijk van de omgeving waarin de component wordt geplaatst en kan in principe in een database worden opgenomen. Het is ook mogelijk bepaalde condities in de formule op te nemen op basis waarvan een actie kan worden uitgevoerd.
Deze acties moeten wel van te voren zijn gedefinieerd (geprogrammeerd).

Kennisacquisitie : vastleggen van de componenten

Om een componentenbibliotheek op te bouwen is het essentieel een aantal niveaus te onderscheiden, waarop met het systeem wordt gewerkt. Met name tijdens het definiëren van de componenten en (sub)systemen spelen deze niveaus een grote rol. Eigenlijk is het werken met het systeem een steeds dieper gaande definitie van de componenten waaruit de installatie bestaat.
Op het laagste niveau is dit de ontwikkelaar of programmeur. Deze levert als het ware alleen de kale bouwstenen met een set gereedschap waarmee de bouwstenen kunnen worden bewerkt. Dit wordt het 'metamodel' genoemd. Het is zelfs nog geen model maar de beschrijving van een model. Met deze bouwstenen maakt de expert een prototype van een component of (sub)systeem. In termen van de object-representatie geeft hij aan welke objecten er bestaan, welke attributen deze objecten hebben en welke afhankelijkheden er tussen de object-attributen bestaan. Dit wordt vastgelegd in de databases Component, Hasparts en Connection. De eindgebruiker zal met deze prototypes werken en ze verder in (laten) vullen.
Het is hierbij de bedoeling, dat de eindgebruiker de belangrijkste beslissingen zelf neemt : het ICAD-systeem vervult zuiver een begeleidende en adviserende taak. Wel is het zo dat bepaalde attribuutwaarden afhankelijk zijn van de context waarin de component is geplaatst. In dit geval is het wèl het ICAD-systeem dat de attribuutwaarde bepaalt. Dit is uiteindelijk ook één van de doelstellingen van het gebruik van het systeem : zoveel mogelijk routinematige werkzaamheden worden door het systeem uitgevoerd. Zodoende kan de eindgebruiker zijn aandacht voornamelijk op het eigenlijke ontwerpen concentreren.
De eindgebruiker geeft aan dat een bepaalde component of een bepaald subsysteem toegevoegd moet gaan worden aan het project. Daartoe kan gekozen worden uit een aantal opties, die met behulp van de 'hasparts', die op dat niveau van het project gelden, worden bepaald. Het is niet mogelijk om een component toe te voegen die niet in het subsysteem thuishoort. De hasparts-database bevat dus strategische kennis : welke kennis mag wanneer gebruikt worden. Er vindt dus gelijk al een adviserende taak van het systeem plaats : er worden een aantal keuzemogelijkheden geboden, waaruit de gebruiker kan kiezen.

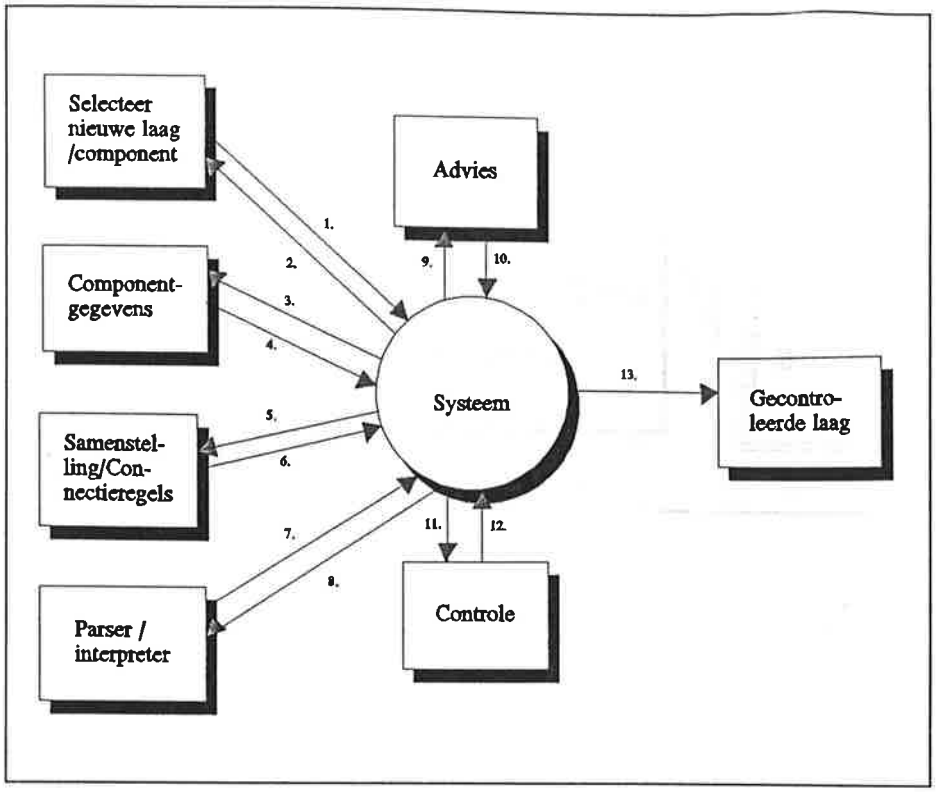
Nadat een component is geselecteerd, wordt de relevante domeinkennis, uit de component-database gehaald. De formules uit deze database worden verwerkt tot bomen met behulp van de parser. Waar mogelijk worden deze bomen al geëvalueerd (dit is mogelijk als er geen variabelen, dat wil zeggen verwijzingen naar waarden van andere slots, in de formule voorkomen). Als er geen waarde of formule in de database staat zal de eindgebruiker deze zelf kunnen toevoegen. Als er geen verwijzingen in andere slots naar dit slot voorkomen, is dit echter niet noodzakelijk.
Nadat een component is toegevoegd, kan deze verbonden worden met andere componenten. Daartoe kan in een database met 'connections' het aantal en de soort verbindingen opgezocht worden. Ook kan het medium (water,lucht) worden bepaald.
Als de eindgebruiker aangeeft, dat op dit niveau alle componenten geplaatst en alle verbindingen gelegd zijn, dan kan het "ontwerpprobleem" worden opgelost. Alle slotwaarden worden (waar nodig en mogelijk) bepaald. Als zich tijdens het berekenen van een slotwaarde een fout voordoet, dan wordt de waarde van het slot leeg gelaten en wordt verder gegaan met de andere slots. Dit betekent echter wel, dat als er een verwijzing naar dit slot voorkomt in een andere formule, dit slot ook niet kan worden berekend zodat een sneeuwbaleffect kan ontstaan.
Pas als alle slots zijn berekend, vindt de definitieve componentselectie plaats. Met de nu bekende waarde van de eigenschappen kan in de database Catalogus worden gekozen uit een aantal componenten die in aanmerking komen om in de installatie te worden geplaatst. Als deze selectie heeft plaatsgevonden, dan is alles bekend over de component en kan deze in de project database worden opgenomen.
In de projectdatabase hoeven de formules volgens welke de bepaling van de slotwaarden worden berekend, strikt genomen niet worden vastgelegd. Men zou kunnen volstaan met het vastleggen van component, fabrikant en typenummer.

Test-systeem "Regelafsluitersysteem"

De installatiekennis van het ICAD systeem bestaat uit de volgende onderdelen:
- Kennis over de componenten;
- Kennis over de samenstelling van een systeem;
- Kennis over de koppeling (connectie).

In figuur 3 is de kennisstructuur van het ICAD-systeem schematisch weergegeven.
In deze afbeelding zijn de volgende informatiestromen gedefinieerd:

1,2. Het selecteren van een nieuwe laag of component.
Wanneer eenmaal een nieuwe laag geselecteerd of toegevoegd is, kunnen er componenten in deze laag worden geplaatst.
3,4. Componentgegevens.
Als een component geselecteerd is, worden hiervan de gegevens opgezocht en opgehaald.
5,6. Samenstelling/Connectieregels.
Bij het selecteren van een nieuwe laag wordt de samenstelling van deze laag opgehaald.
Indien er koppelingen aange-



Figuur 3. Schematische weergave kennisstructuur

bracht worden, wordt gebruik gemaakt van de connectieregels.

7,8. Parser/Interpreter.
Voor het bepalen van waarden gebruikt.

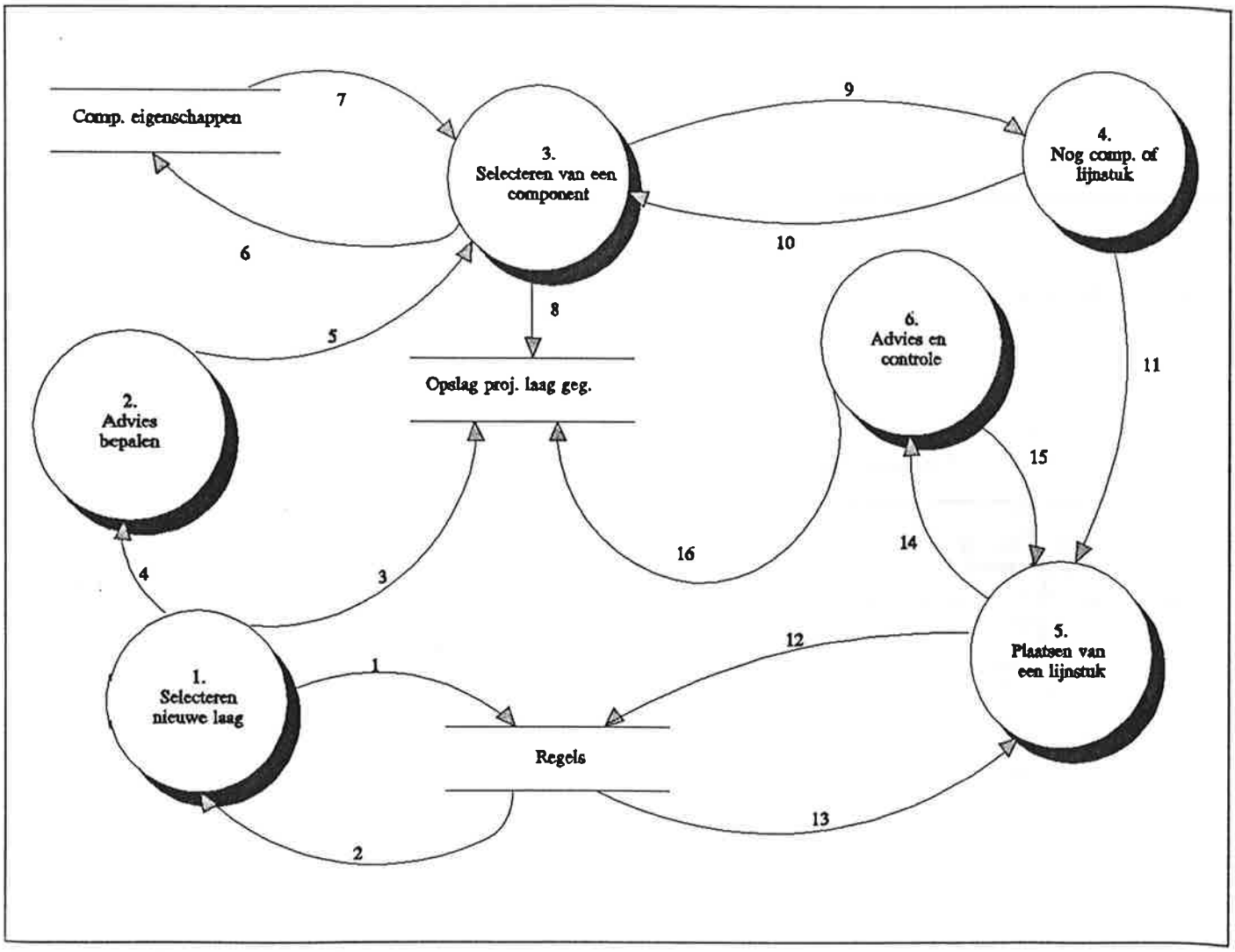
9,10. Advies.
Welke componenten en welke connectie.

11,12. Controle.
Controle vindt plaats op de samenstelling en de connecties van het systeem.

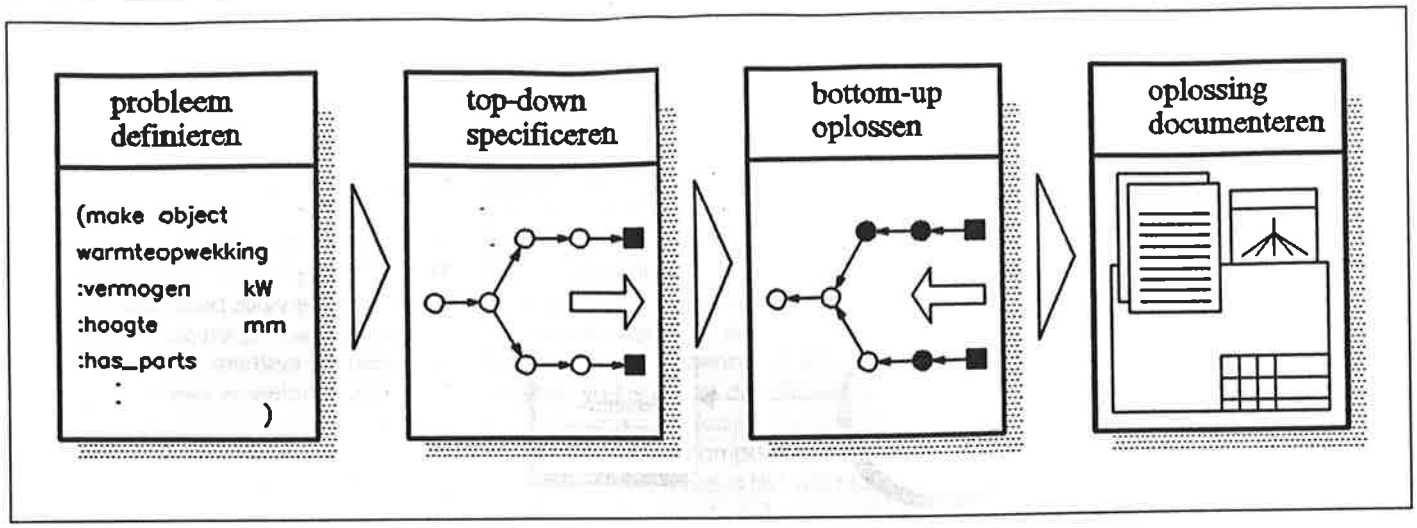
13. Gecontroleerde laag.
De uitvoer is een gecontroleerde laag.

De stromen kunnen ook in een Data Flow Diagram worden weergegeven, zie figuur 4:

Bij het invoeren van een nieuwe laag worden verwijzingen gelegd naar de bijbehorende regels uit een bestand (1,2). Bepaalde gegevens van deze laag kunnen op dit moment al worden opgeslagen, bijvoorbeeld: naam van de laag (3).



Figuur 4. Data Flow Diagram CAD-Systeem



Figuur 5. Redeneerstructuur

Uit de samenstellingsregels zal een advies worden samengesteld (4). Dit advies wordt weergegeven in de vorm van een menu. Aan de hand van het menu bestaat de mogelijkheid om componenten te selecteren (5). Bij elke geselecteerd component horen verschillende eigenschappen (6,7). Als alle componenten ingevoerd zijn, kan er begonnen worden met het trekken van lijnstukken tussen de verschillende componenten (9,10,11). Bij het plaatsen van een lijnstuk wordt er direct gecontroleerd, aan de hand van de regels, of dit een geldige connectie is (12,13,14). Zonodig wordt er een advies gegeven (15). De projectgegevens worden opgeslagen in een bestand (16).

De gebruiker zal worden ondersteund door het ICAD-systeem, doordat hij tijdens de engineering slots kan invullen of kan laten invullen met behulp van de in het systeem beschikbare kennis. Het ontwerpproces krijgt een top-down karakter door de invoer van componentselectie. Elk component heeft haar eigenschappen in de vorm van een waarde of een formule.
Door het interpreteren van de kennisregels en de formules, door het ICAD-systeem, ontstaat er een bottom-up bewerking. Dit betekent dat een eigenschap van een component in de vorm van een formule bottom-up berekend wordt.

Deze redeneerstructuur is in fig. 5 weergegeven.

Als in een formule variabele eigenschappen van een component voorkomen, kunnen deze zich in een slot van hetzelfde component of van een ander component bevinden. Ontbrekende informatie zal aan de gebruiker worden gevraagd. Zo zal het ICAD-systeem doorgaan tot alle benodigde gegevens zijn verzameld en de resulterende waarde in het slot is ingevuld.

Als testsysteem is gekozen voor een luchtbehandelingssysteem als in figuur 6 weergegeven.

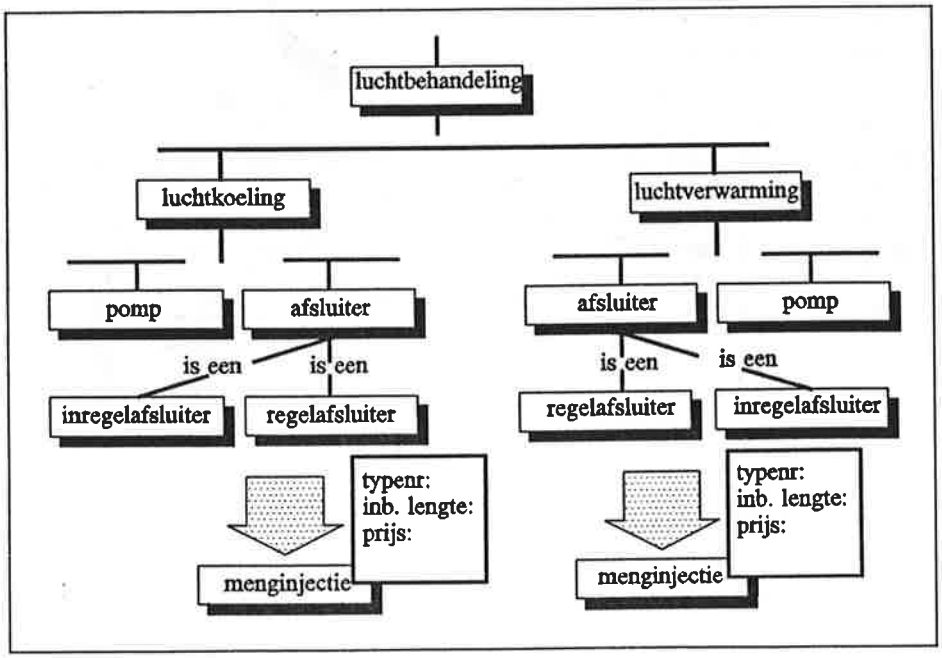
Dit is een semantisch netwerk dat door steeds verder gaande verfijning eigenschappen bepaalt.

Doorrekenen regelafsluitersysteem

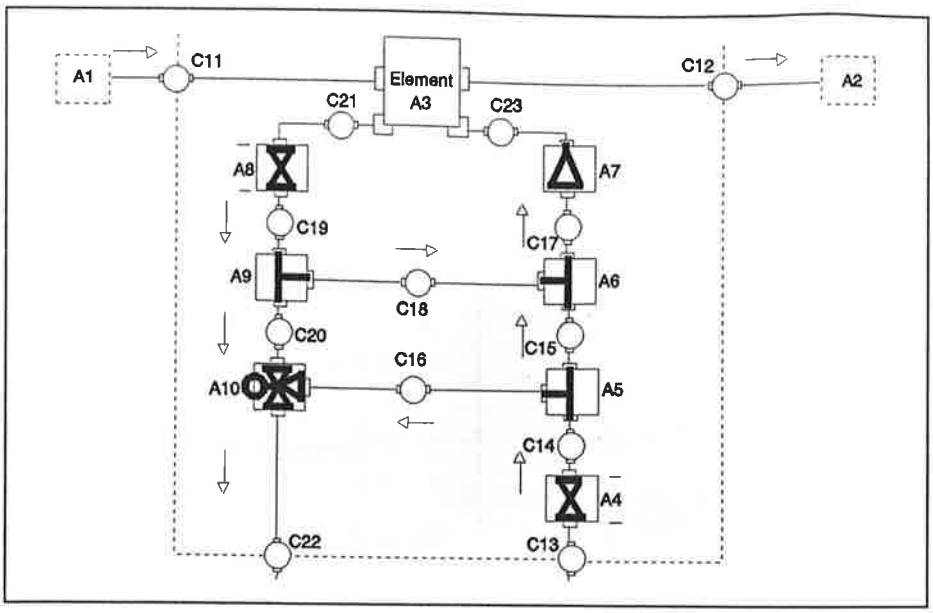
De gebruiker heeft de keuze van een systeem gemaakt. Hieronder wordt als voorbeeld een menginjectie gehanteerd. In het regelafsluitersysteem bevinden zich een aantal vaste componenten. Deze worden in de data-structuur weergegeven met behulp van 'atoms'. Zoals hiervoor reeds is vermeld is een atom de representatie van een component op basis van het objectgeoriënteerde principe. Elk atom heeft slots waarin de eigenschappen zijn opgeslagen. Elke component is verbonden met andere componenten door middel van connecties.

Deze connectie bevat slots met eigenschappen zoals het medium, de volumestroom en de lengte van de leiding. Elke connectie en elk atom heeft twee connectiepunten. In een connectiepunt bevindt zich:
- Het slot met de stroomrichting;
- Een slot met het medium (indien nodig).

Bij het leggen van de connecties tussen



Figuur 6. Luchtbehandelingssyteem



Figuur 7. Verbindingsstructuur testsysteem

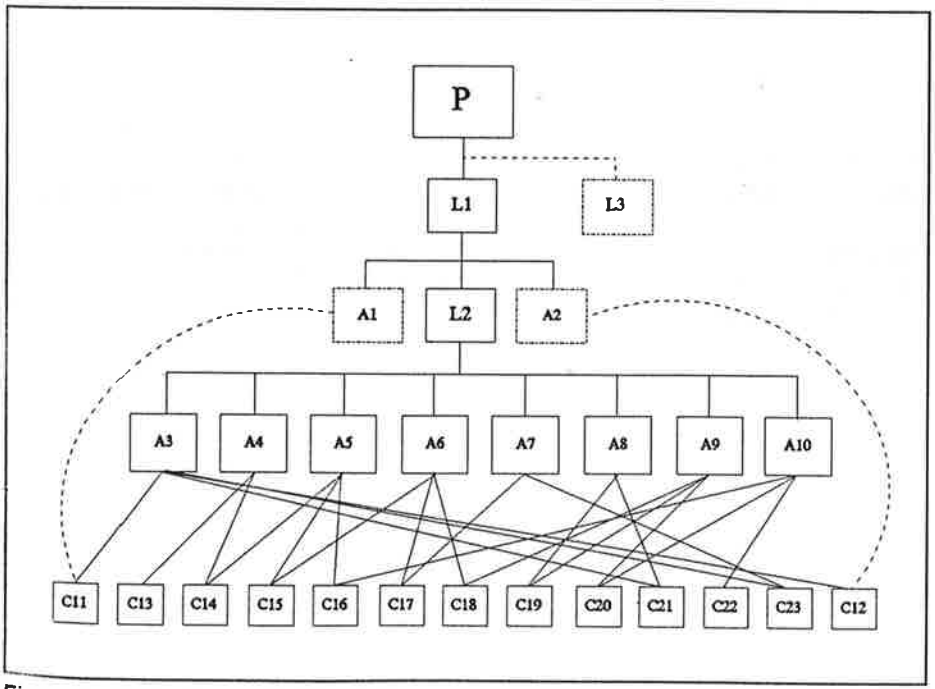
de atoms moet eerst worden bekeken of de connectie kan worden gelegd. Dit is afhankelijk van de grootte van de volumestroom, het medium (lucht, stroom, electriciteit, warm of koud water) en de richting. Is de connectie geldig, dan wordt deze geplaatst. Bij het gekozen menginjectiesysteem is de opbouw altijd hetzelfde. Dit heeft als voordeel dat de structuur in zijn geheel in de database kan worden vastgelegd.

De samenstelling staat hierdoor vast en de connecties zijn geplaatst. Alle eigenschappen van de atoms worden uit de database opgehaald en de rest van de benodigde gegevens wordt verzameld.

Het project dat bestaat uit een luchtbehandelingssysteem is opgebouwd zoals in figuur 8 is weergegeven.

In figuur 8 is:
L1 = Luchtbehandelingssysteem;
L2 = Meng-injectiesysteem.

Bij het doorrekenen van een bepaalde laag, is het nodig om langs de componenten te "reizen", met behulp van een 'travel'. Deze procedure zoekt een slot op. De travel begint altijd bij een atom met ingaande stroom die een connectie heeft die verbonden is met een andere laag. Is deze atom gevonden dan loopt men alle atomconnecties langs. Per atom en connectie worden alle slots doorlopen, zodat de eigenschappen kunnen worden berekend.



Figuur 8. Verbindingsstructuur project

De volgende eisen worden aan de travel gesteld:
- Alle atoms die worden doorlopen door de travel, moeten in dezelfde laag zitten;
- Er moet voor gezorgd worden dat de travel de juiste weg doorloopt, dat wil zeggen met de stroomrichting mee;
- Één keer een bepaalde weg doorlopen is voldoende.

Een atom kan connecties met meerdere media hebben. Dit geeft problemen met het doorlopen van de travel in zo'n atom. Voor dit probleem is de volgende oplossing gevonden.

Alle atoms en connecties krijgen connectiepunten. Bij een atom waar meerdere media voorkomen heeft elk invoerkanaal en uitvoerkanaal een connectiepunt met daarin vastgelegd het medium en de stroomrichting. Zo kan er door de travel getest worden op het medium en de stroomrichting en afhankelijk daarvan de juiste weg worden genomen.

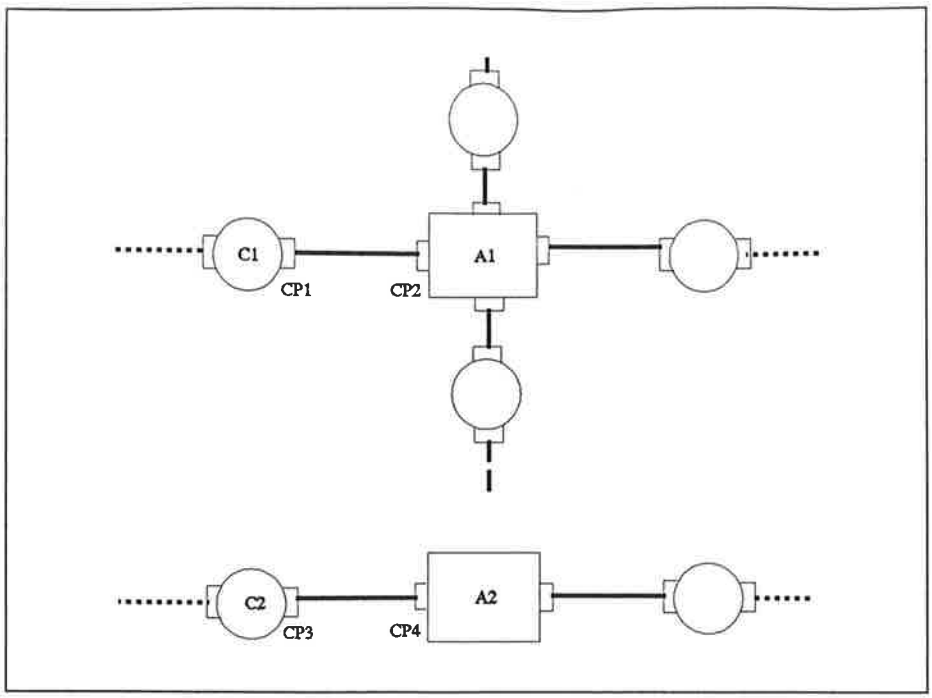
Atom met meerdere media:
A1 => Atom eigenschappen, naam
C1 => Connectie eigenschappen
CP2 => Stroomrichting, medium

Atom met één medium:
A2 => Atom eigenschappen, naam, medium
C2 => Connectie eigenschappen
CP1,3,4 => Geen stroom richting

In slots van atoms en connecties bevinden zich de eigenschappen. Zo'n eigenschap kan bestaan uit een formule of een regel. Een formule kan variabelen bevatten. Deze variabelen kunnen eigenschappen van dezelfde component of van een andere component zijn.

Een variabele die zich bevindt in het slot van dezelfde component wordt snel gevonden. Met de naam van de variabele wordt er in de slotcontainer van dat bepaalde atom gekeken en een eigenschap gevonden met die variabele naam.

Bevindt de variabele zich bij een ander atom in de slotcontainer dan blijkt het niet eenvoudig. Er wordt dan bottom-up gezocht naar de variabele naam. Eerst in de omliggende atomen in dezelfde laag en daarna in de bovenliggende lagen.



Figuur 9. Opslag medium & media

Evaluatie van het systeem

Voor zover nu te overzien is, is het goed mogelijk de eindgebruiker tot op een laag niveau van het installatie-ontwerp (dat wil zeggen gedetailleerd) te ondersteunen. Vaak zal een aantal alternatieven kunnen worden aangeboden: de eindgebruiker beslist in deze gevallen wat er moet gebeuren.

De opzet van het ICAD-systeem is flexibel genoeg om een groot aantal verschillende manieren van ontwerpen mogelijk te maken. Dit komt ondermeer doordat het gebruik van databases om de component-eigenschappen vast te leggen, het mogelijk maakt dat een eigenschap wordt toegevoegd of gewijzigd.
Met het systeem is het mogelijk om een zeer globaal ontwerp te maken, dat pas later wordt uitgediept. Dit is bij het maken van een voorlopig installatie-ontwerp wel zo prettig, het is dan nog niet noodzakelijk om alle details mee te nemen.

Literatuurlijst

1. Amerongen K. van, Vernooij I., Kennis in een CAD-systeem, afstudeerverslag Hogeschool Utrecht, afdeling Technische Informatica, 21 mei 1992.
2. Struck, D. Konzeption und Entwicklung einer wissenbasierten Konstruktions und Planungsumgebung, dissertatie aan de Rheinisch Westfälischen technischen Hochschule, Aachen 1989.
3. Koning P. de, Winnips F.J., Zeiler W., (1991), Object Orientatie in computer ondersteund ontwerp, Kennissystemen nr. 4. 1991.
4. Winnips F.J., (1991) Ontwerpen en kennissystemen in de installatietechniek, OC-D-200 afstudeerverslag Universiteit Twente, faculteit der Werktuigbouwkunde vakgroep ontwerp- en constructieleer. Enschede 1991.
5. Winnips F.J., Koning P. de, Zeiler W., Een intelligent ontwerpsysteem, NVKI-Nieuwsbrief nr. 2. april 1992.
6. Zeiler W. (1991) Towards real ICAD-systems with AI technology, proceedings, G. Doumeigts, J. Browne and M. Tomljanovisch (eds.), Computer Applications in Production and Engineering, (Proc. 4th Int. I.F.I.P TC5 Cont. on Integration Aspects, CAPE '91, Bordeaux, France september 1991), North-Holland, Amsterdam.
7. Zeiler W. (1992) Towards a real Cad-system using AI-technology, Proceeding Fifth International Conference on Industrial and Engineering Applications of Artifical Intelligence and Expert Systems (IEA92AIE) Paderborn.

Toelichting: Object Oriented Programming (OOP)

Inleiding

Het doel van deze toelichting is een algemeen overzicht te geven van het ontstaan, gebruik en van de filosofie van een Object Georiënteerde Programmeertaal (OOP). De gebruikte Object Oriented programmeertaal binnen Kropman was tot voor kort C++ van Zortech. Maar men is overgegaan naar Borland C++. Dit i.v.m. de betere mogelijkheden voor het schrijven van applicaties 'onder' windows.

Ontwikkeling van C++

De programmeer taal C is een van de meest gebruikte programmeertalen. Het is een flexibele en zeer krachtige programmeer taal, C werd gebruikt voor de ontwikkeling van de meeste belangrijke softwareprodukten voor de laatste jaren. Desondanks heeft C ook zijn beperkingen. Als een programma te groot wordt, wordt het moeilijk te onderhouden omdat het niet eenvoudig als een geheel te omvatten is. Om dit probleem te ondervangen werden er een aantal extensies aan C toegevoegd. Deze nieuwe taal stond in de begin jaren bekend onder de naam 'C with Classes'. Deze naam werd vervolgens veranderd in 'C++'. Deze toevoegingen ondersteunen OOP.

Wat is Object Oriented Programming?

Het doel van Object Oriented Programming (OOP) is de programmeur de mogelijkheid te geven om grote en complexe programma's beter te beheren en te begrijpen. OOP geeft de mogelijkheid om eenvoudiger complexe problemen op te delen in subproblemen. Een afzonderlijk subprobleem wordt een object genoemd. Alle OOP talen hebben drie dingen met elkaar gemeen:
- Objects
- Polymorphism
- Inheritance

Objects

Een object is een logische eenheid die zowel data als code, die de betreffende data manipuleert, bevat. Binnen een object kan een gedeelte van de data en/of code alleen door het betreffende object worden benaderd. Deze data en code behoren dan tot de private elementen van het object, en zijn niet toegankelijk voor andere code buiten het betreffende object. Op deze manier geeft een object een hoge graad van protectie tegen andere delen van het programma die per ongeluk gebruik maken of veranderen van private elementen van een object. De koppeling van data en de code die deze data manipuleert wordt 'encapsulation' genoemd.

Polymorphism

Polymorphism betekent dat één naam voor een aantal samenhangende maar lichtelijk afwijkende taken kan worden gebruikt. Het voordeel hiervan is om het programma, aan de hand van het meegegeven data type, zelf te laten beslissen welk gedeelte code moet worden uitgevoerd.

Inheritance

Inheritance of overerving is het proces waarin een object de eigenschappen

van een ander object kan krijgen. De meeste kennis wordt handelbaar gemaakt door deze te splitsen in hiërarchische 'classes'. Bijvoorbeeld een appel hoort onder de meer omvattende 'class fruit', deze hoort onder de meer algemene class: eten. Zonder inheritance zouden alle objecten zeer expliciet en in alle facetten moeten worden gedefinieerd. Door gebruik van classes echter is het mogelijk om een object alleen maar te definiëren door dié elementen die hem uniek maken binnen zijn huidige class en de andere meer algemene elementen die het object deelt met zijn algemene class, te erven.

Woordenlijst

Abstract Datatype :	het gebruik van data type (v.b. klasse) zonder kennis over de implementatie.
Atom :	implementatie van het metamodel in de vorm van een object.
Auto-CAD :	teken/ontwerp programma.
Btrieve :	Recordmanager van Novell.
CAD :	Computer Aided Design.
Call :	procedure-aanroep.
Compileren :	vertalen van een programma van programmacode naar machinecode.
Connection :	aansluitpunt, hierin wordt de informatie over de verbinding opgeslagen.
Decompositie :	uitsplitsing in delen.
DLL :	Dynamic Link Library.
DOS :	Disk Operating System.
DPMI :	Dos Protected Mode Interface.
Dynamic Binding :	dit mechanisme kiest tijdens run-time welke methode gebruikt wordt.
Frame :	raamwerk om fragmenten van kennis te groeperen.
GUI :	Graphical User Interface.
Hybride :	uit heterogene elementen bestaand.
Information Hiding :	een regel voor data toegang. Alleen die procedures krijgen toegang die speciaal voor deze data bestemd zijn.
Inferentiemechanisme :	mechanisme waarmee een probleem wordt opgelost.
Interperter :	een programma dat er voor zorgt dat een gecompileerd programma uiteindelijk ook echt kan worden uitgevoerd.
ICAD :	Intelligent computer ondersteunend ontwerpprogramma.
Inheritance :	het mechanisme ervan is een tranfer van bezittingen van een klasse naar een subklasse.
Layer :	de structuur van de installatie is opgebouwd uit verschillende niveaus. In deze class wordt de hierarchie van de installatie bewaard.
Library :	bibliotheek : groepering van verschillende zaken.
Linken :	het aan elkaar koppelen van verschillende onderdelen van een programma.
Metamodel :	overkoepelend model waarin opgenomen de totale structuur van het gehele datamodel.
O.A.V. :	Object Attribute Value.
Object :	datastructuur waarin zowel datavelden of kennis als methoden worden gegroepeerd.
O.O. :	Object Georiënteerd.
O.O.D. :	Object Georiënteerd ontwerp.
OOP :	Object Oriented Programming.
OOPL :	Object Oriented Programming Languages.
Overlay :	gedeelte van een programma dat tijdens het uitvoeren van een programma pas in het geheugen wordt geladen.
Overloading :	Operator "Over loading" is een principe dat gebruikt kan worden voor het uitvoeren van dezelfde operaties op data met verschillende typen van de parameters (bijv. real en integer).
Parser :	methode die iedere keer een element uit de formulestring leest.
Parsen :	het opbouwen en doorrekenen van productiebomen vanuit een formulestring, het proces dat een lineaire representatie structureert aan de hand van een gegeven grammatica.
Paradigma :	lay-out, raamwerk voor manier van werken.
Protected-mode :	processormode waarbij de uitgebreidere mogelijkheden van de intel-processor vanaf de 80286 worden benut.
Real-mode :	processormode waarbij de processor werkt als een normale 8088/8086 processor.
Semantiek :	betekenis.
Slotcontainer :	container die de eigenschappen van een object bevat.
String :	een eindige rij symbolen.
Triplet :	stel van drie bijeenhorende zaken.
Terminal :	een "eindsymbool" dat geen symbolen kan generen.