SERIE RESEARCH MEMORANDA

PRINCIPES EN GEBRUIK VAN ENVISAGE

E.R.K. SPOOR
J.W.B. VERMEULEN

Research Memorandum 1988-35 augustus 1988

VRIJE UNIVERSITEIT
FACULTEIT DER ECONOMISCHE WETENSCHAPPEN
EN ECONOMETRIE
A M S T E R D A M

PRINCIPES EN GEBRUIK VAN ENVISAGE

door

E.R.K. Spoor &
J.W.B. Vermeulen

Vrije Universiteit
Faculteit der Economische Wetenschappen
en Econometrie
AMSTERDAM

ABSTRACT

Een knowledge engineer zonder hulpmiddelen is als een timmerman zonder gereedschap. Het kiezen van een geschikt hulpmiddel wordt bemoeilijkt door enerzijds de veelheid aan beschikbare produkten en anderzijds de complexiteit van die produkten. Zaak voor de knowledge engineer om zich goed te orienteren.
Dit artikel beschrijft de principes van, en enige ervaring met één zo'n hulpmiddel, de expertsystem-shell ENVISAGE[1].

INLEIDING

In een hoog tempo groeit de belangstelling voor het fenomeen expertsysteem. Die belangstelling wordt enerzijds gewekt door al of niet gegronde verwachtingen van gebruikerszijde omtrent de toepassingsmogelijkheden van expertsystemen, anderzijds door de 'push' die leveranciers van hulpmiddelen op dit terrein (expertsystem-shell's, ontwikkelomgevingen, e.d.) bewerkstelligen. Aanbod schept ook hier de vraag wellicht.
Zoals het vaker is gegaan in het verleden, heeft zo'n snelle ontwikkeling ook een schaarste aan expertise, dat wil zeggen, aan 'knowledge engineers', tot gevolg. Als je één of twee expertsysteempjes hebt gebouwd, ben je meteen een 'crack' en éénoog in een land van blinden. Maar enige introspectie leert de betrekkelijkheid hiervan en de volgende conclusie is dat het niet zo eenvoudig is een expertsysteem van enige betekenis te maken, ook niet als er geavanceerde hulpmiddelen beschikbaar zijn.
Voor het definiëren, ontwerpen en realiseren van een 'klassiek' informatiesysteem zijn in de loop der tijden methoden en technieken ontwikkeld en weet iedereen in gescheiden functies (analist, ontwerper, e.d.) wat er dient te gebeuren. Doch het bouwen van een expertsysteem wordt, ofschoon een erkend complex probleem, vooralsnog op de schouders geladen van één enkele knowledge engineer, die dan maar alles moet kunnen. Acquisitie en identificatie, conceptualisatie, formalisatie, implementatie ...[1], voor alle aspecten dient hij over vaardigheden en technieken te beschikken. Geen wonder dat shell's en ontwikkelomgevingen zo'n furore maken. Zonder geschikt gereedschap is het bijna geen

---

[1] ENVISAGE is een produkt van Systems Designers B.V., Zeist

haalbare kaart meer.
Voor de knowledge engineer is het zaak te weten wat er te koop is op het gebied van hulpmiddelen ten behoeve van de bouw van expertsystemen en hoe deze middelen kunnen worden gebruikt.

Dit artikel beschrijft, aan de hand van een niet te klein, maar ook niet al te complex probleem hoe één van die hulpmiddelen, de shell ENVISAGE, werkt en ervaren is.
De indeling van de hoofdstukken is als volgt. Allereerst worden de belangrijkste aspecten van Envisage belicht aan de hand van enkele kleine voorbeelden (hoofdstuk 1). Vervolgens wordt in hoofdstuk 2 beschreven binnen welk kader het produkt getoetst is, dat wil zeggen de probleemomgeving. Hoofdstuk 3 gaat in op de fasegewijze ontwikkeling van het proefsysteem, waarna tenslotte ervaringen met Envisage, gegeven de proefomgeving, worden beschreven in hoofdstuk 4.

1. Over envisage

Allereerst worden enkele algemene eigenschappen van Envisage beschreven. Vervolgens komen de belangrijkste componenten aan bod. De segmentatie van complexe modellen wordt afzonderlijk belicht, evenals de besturing van het redeneermechanisme. Tenslotte zal aandacht worden besteed aan de verschillende datatypen en -strukturen.

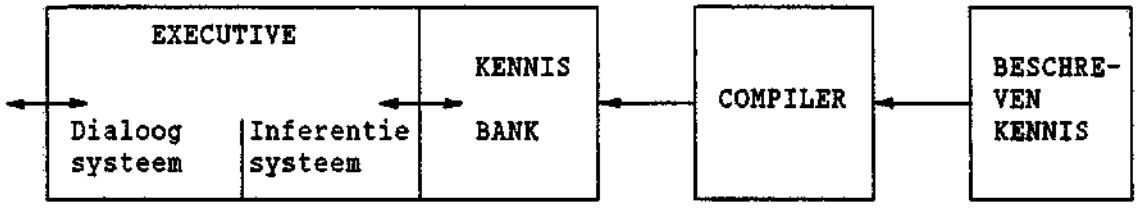
1.1 Algemene eigenschappen

Envisage is een zogenaamd 'rule-based' expert-systeem. Alle kennis wordt vastgelegd in regels zoals:

beeldbuis_stuk INDIEN weinig_licht EN toestel_oud,

waarin beeldbuis_stuk, weinig_licht en toestel_oud variabelen zijn die een conditie in de werkelijkheid beschrijven. De regel geeft een oorzakelijk verband tussen die variabelen aan.
Envisage gebruikt de techniek van 'backward-chaining' om m.b.v. de regels te redeneren; indien de juistheid van een waarde van een variabele moet worden vastgesteld, zal het expertsysteem in zijn bestand met regels zoeken naar een regel die bewijs levert voor de waarde. Dit proces zet zich recursief voort tot variabelen worden aangetroffen welke blijkbaar een elementair ervaringsfeit weergeven, waarvan de waarden middels een dialoog gevraagd mogen worden. In het voorbeeld

moet voor beeldbuis_stuk de waarde van de variabelen weinig_licht en toestel_oud vastgesteld worden. Weinig_licht zou direct gevraagd kunnen worden, en toestel_oud wellicht weer afgeleid worden uit het aantal kanalen, tiptoetsen of niet e.d.
Duidelijk is dat, alvorens Envisage kan redeneren, een kennisbank opgesteld moet worden. Dit geschiedt in een speciaal taaltje, waarvan de essenties hieronder beschreven zullen worden.
Behalve de eigenlijke regels, kunnen ook de wijze van vragen-stellen en de volgorde van redeneren, c.q. redenatiedoel gespecificeerd worden. Direct interpreteren van de beschreven kennis zou vertragend werken zodat een compiler noodzakelijk, maar gelukkig ook voorhanden is. Het benaderen van de vertaalde kennis, het redeneren en het sturen van de dialoog komt voor rekening van de executive.
Schematisch is dit als volgt weer te geven:



1.2 De belangrijkste kenniselementen

Een kennisbank in Envisage onderscheidt als belangrijkste kenniselementen:

1. feiten zoals asserties (beweringen), objecten en waarden;
2. verschillende soorten regels;
3. vragen en
4. acties ter sturing van het inferentiesysteem.

Het volgende voorbeeld en de aansluitende verklaring van de gebruikte kenniselementen geven een idee van de opbouw van een kennisbank.

```
MODEL beeldbuis
VERSION 1.0
INTRO "Model om conditie beeldbuis vast te stellen"

ASSERTION beeldbuis_stuk: "is beeldbuis versleten?"
ASSERTION weinig_licht: "weinig licht in buis?"
```

```
ASSERTION toestel_oud: "oude bak?"

QUESTION beeldbuis_vragen: "vragen m.b.t. beeldbuis"
OBTAIN weinig_licht
  WITH  "Projecteer een testbeeld op de t.v. Zit er weinig",
    "contrast en/of licht in, en is dit ook niet middels",
    "de helderheid- en contrastknoppen op te heffen?"
  WHY "Misschien is de beeldbuis wel versleten."
  YESNO
ALSO OBTAIN toestel_oud
  WITH "Is het toestel ouder dan zes jaar?"
  WHY "Als het toestel jonger is kan de beeldbuis niet",
    "stuk zijn."
  YESNO
RULE beeldbuis_rules: "regels m.b.t. beeldbuis"
beeldbuis_stuk IS weinig_licht AND toestel_oud

ACTION beeldbuis_doen: "bestuur consultatie"
  CONSIDER beeldbuis_stuk
ALSO ADVISE "!N!NUw vermoeden dat de beeldbuis versleten is,",
      "is voorzover ik kan bepalen: ",beeldbuis_stuk
```

Model-identificatie

De MODEL, VERSION en INTRO declaraties identificeren het model, zoals in een COBOL-programma de 'Identification Division' dit doet. De tekst achter INTRO wordt afgedrukt op het scherm om de gebruiker in te lichten over de aard van het onderhavige model.

ASSERTION-declaratie

Een assertie beschrijft een variabele van het type 'boolean', met dien verstande dat FALSE en TRUE de extremen van een continuüm vormen in verband met de mogelijkheid tot het redeneren met onzekerheid. Ook andere typen variabelen dan asserties kunnen beschreven worden, zie hiervoor par. 1.5.

Na het ASSERTION-sleutelwoord wordt de naam van de variabele genoteerd, gevolgd door een dubbele punt en een verduidelijkende string. Deze string is in bijna iedere instruktie verplicht, doch heeft maar zelden

een aantoonbare functie.
Na een variabele-declaratie zoals weergegeven, zijn er nog allerlei declaraties mogelijk betreffende de opmaak bij uitvoer van het item (b.v. in plaats van de getalswaarde, een omschrijving uit een schaal (25° → "lauw") en andere, minder essentiële zaken).

QUESTION-declaratie

Het sleutelwoord QUESTION geeft een cluster van vragen aan, waarin stukjes dialoog gespecificeerd worden. Als het expertsysteem voor een bepaalde variabele een QUESTION beschikbaar heeft, kan de gebruiker gevraagd worden de waarde van die variabele in te voeren. Indien er geen QUESTION voor die variabele aanwezig is, zal het systeem de waarde trachten vast te stellen door terugwaarts te redeneren. Naast het op deze wijze 'vraagbaar' maken van een variabele is het ook mogelijk om bij declaratie van de variabele het attribuut ASKABLE mee te geven, wat voor ENVISAGE een aanwijzing is dat de variabele gevraagd mag worden, en dat het systeem daarbij zelf een vraag in elkaar mag knutselen. Zeker in het nederlands leidt dit tot primitieve grammaticale constructies, zodat expliciet gebruik van QUESTION aan te bevelen is.
Het doel van het samenbundelen van een aantal vragen in een QUESTION-cluster is om gemeenschappelijke PROVIDED- en WHY-declaratie mee te kunnen geven. PROVIDED geeft aan, dat er een gemeenschappelijke voorwaarde gesteld kan worden voordat één van de vragen gesteld wordt. De WHY-declaratie op QUESTION-niveau declareert tweede orde WHY-tekst. Als de gebruiker WHY intikt als antwoord op een vraag, dan wordt eerst de WHY-tekst van die vraag afgedrukt. Door nogmaals WHY in te geven, volgt de WHY-tekst van de cluster.

RULE-declaratie

Evenals vragen worden ook regels geclusterd, middels een zogenaamde 'RULE-sentence'. In het voorbeeld is er evenwel slechts een regel in de cluster aanwezig. Ook hier is het weer mogelijk om een gemeenschappelijke PROVIDED op te geven.
Binnen rules zijn tal van expressies mogelijk, en een groot aantal standaardfuncties is voorhanden. Ook kunnen PASCAL-subroutines aangeroepen worden. Envisage doet hier niet onder voor de faciliteiten van de gemiddelde programmeertaal.

ACTION-declaratie

Een ACTION is alweer te zien als een cluster, ditmaal van besturingsinstrukties. Op deze wijze is het mogelijk voor alle besturingsinstrukties een gemeenschappelijke PROVIDED mee te geven. Een ACTION heeft wel iets weg van een subroutine. Bij executie van het model worden alle ACTION's (op modelniveau) op de 'goal-stack' gezet om uitgevoerd te worden, waarna de eerste ACTION wordt benaderd. Is er een PROVIDED dan wordt deze eerst getest, voordat alle instrukties binnen de ACTION op de goal-stack gezet worden. Vervolgens worden deze instrukties uitgevoerd, en zet het proces zich voort. Overigens kunnen aan deze instrukties opties als URGENT en LATER toegevoegd worden, om de volgorde van redenatie en consultatie te veranderen.
De twee belangrijkste besturingsinstrukties zijn CONSIDER en ADVISE. CONSIDER wil zoveel betekenen als 'stel de waarde vast van ...'. Deze instruktie zet de redeneermachine in gang. De tweede instruktie, ADVISE doet niets meer dan het afdrukken van de lijst van strings en variabelen die erachter staat, en wordt dus gebruikt om de adviezen e.d. af te drukken.
Andere besturingsinstrukties worden in 1.4 beschreven.

1.3 Segmentatie van het model

Het is duidelijk dat een model van enige betekenis aanzienlijk meer tekst zal beslaan dan het voorbeeld, en dat het overzicht in schier onafzienbare reeksen vragen, regels, variabelen en acties al gauw verloren dreigt te gaan. Het is dus noodzakelijk om het model te segmenteren, en om tot min of meer onafhankelijke subsystemen te komen.
Envisage kent hiertoe zogenaamde AREA's. Een area kan men het beste zien als een submodel compleet met variabelen, regels vragen en acties. De CONSIDER-instruktie roept een area aan, wat tot gevolg heeft dat alle acties binnen de betrokken area uitgevoerd zullen worden.
Het inferentiesysteem zal in principe alleen objecten binnen de actieve area benaderen, en niet kijken naar bijvoorbeeld vragen en regels in andere areas of op modelniveau. Ook variabelen in andere areas of op modelniveau blijven afgeschermd, tenzij ze expliciet binnen de betrokken area worden geïmporteerd (en dan uitsluitend 'lezend' beschikbaar zijn).

Behalve leesbaarheid en onderhoudbaarheid van het programma heeft deze opdeling in areas tevens het gevolg, dat meer efficiente executie van het model mogelijk is, immers, er hoeft in een beperktere regio naar regels en naar vragen gezocht te worden.

1.4 Besturingsinstrukties

Als besturingsinstrukties hebben we, in het voorbeeld in par. 1.2, ADVISE en CONSIDER gezien. Hoewel deze, gecombineerd met het inferentiesysteem al een behoorlijk krachtige omgeving creëren (zo krachtig zelfs dat binnen het proef-systeem (zie hoofdstuk 3) geen andere instrukties gebruikt behoeven te worden) zijn er toch allerlei instrukties nodig om de consultatie te besturen.
Deze instrukties kunnen op drie plaatsen voorkomen:

- binnen ACTIONs. Bij uitvoer van het model of van een area daarvan, worden alle betreffende acties uitgevoerd, als de PROVIDED voor die actie tot 'true' evalueert. Zoals we hebben gezien, kan een actie een of meerdere instrukties bevatten;
- binnen DEMON's. Deze lijken op acties maar worden normaal niet uitgevoerd. Aan ieder demon zijn een of meerdere voorwaarden gekoppeld, en zodra aan een van die voorwaarden voldaan wordt, worden de instrukties in de betreffende demon uitgevoerd. Een demon hangt de normale executie als een zwaard boven het hoofd, klaar om toe te slaan. Een toepassing zou bijvoorbeeld kunnen zijn: inputvalidatie (DEMON WHEN geboortejaar > sterfjaar);
- tijdens de executie. In plaats van op een vraag te antwoorden kan op een speciale instruktieregel van het beeldscherm, een opdracht worden ingegeven, die onmiddellijk wordt uitgevoerd.

Alle besturingsinstrukties kunnen, zij het soms met een omweg, in ieder van de drie situaties gebruikt worden. We zullen nu een kort overzicht van de beschikbare instrukties geven, zonder volledigheid te betrachten:

- besturing van de werkruimte: WIPE, RECONSIDER, RESTORE, SAVE, om gedeelten van de werkruimte uit te wissen, opnieuw te bepalen of de gehele werkruimte te laden of op schijf op te slaan. BOTHER, om fout ingevoerde variabelen opnieuw in te kunnen voeren.
- advies: ADVISE, RECORD, BROADCAST, om teksten naar het beeldscherm te sturen, in een adviesbestand op te slaan, of beide. REPORT, om de adviesbestand op het beeldscherm weer te geven.

- registratie: LOG en CLOSELOG, om de consultatie in zijn geheel vast te leggen.
- toelichting: WHY, om why-teksten af te drukken, EXPLAIN, om uitgebreidere uitleg te krijgen indien beschikbaar, HELP, voor algemene hulp, FACTORS, om redeneerpaden weer te geven, TRACE, om het verloop van de redenering te kunnen volgen.
- redenering: CONSIDER, om de waarde van een variabele middels redeneren en vragen te bepalen, EVALUATE, idem, maar zonder verdere vragen te stellen. OFFER, om waarden van variabelen in te kunnen voeren zonder dat er een vraag gesteld wordt (waardoor 'pseudo-forward-chaining' bereikt kan worden).
- besturing: BEGIN, om de consultatie opnieuw te beginnen, MODEL, om een model in te laden, DO, om een 'script' van commando's (in een bestand) uit te laten voeren en CALL, om PASCAL-procedures aan te kunnen roepen.

1.5 Datatypen en -structuren

Zoals al gezegd vindt representatie van kennis in Envisage plaats via regels. Om met deze kennis te kunnen redeneren, zijn instantiaties, gegevens nodig welke in variabelen opgeslagen worden. Deze gegevens kunnen zich in tal van gedaanten aandienen, zoals getallen, strings, zekerheidspercentages e.d.. Ook combinaties tot complexere structuren zijn denkbaar.
We zullen het arsenaal van mogelijkheden om gegevens adequaat te beschrijven in Envisage inventariseren, waarbij we ons eerst richten op de beschikbare elementaire datatypen, en vervolgens op de complexere structuren die hieruit opgebouwd kunnen worden.

Elementaire datatypen:

ASSERTION	een zekerheidpercentage, of zwart/wit gezien, ja/nee variabele. In- en uitvoer is mogelijk middels schalen van -5..5, 0..1 of n:1 (b.v. 4 tegen 1 dat het zo is). Ook is eenvoudigweg YES/NO mogelijk.
INTEGER	een geheel getal tussen -32768 en 32767. Invoer is mogelijk via de numerieke getalswaarde of via een menu. Uitvoer is mogelijk als getal of als label uit een schaal (b.v. 20..29 wordt vertaald in 'lauw' als het

temperatuur van water betreft).

STRING — een reeks karakters van willekeurige lengte. Hierin kunnen stukjes tekst, b.v. namen opgeslagen worden.

OBJECT — als INTEGER, echter nu een floating-point getal waarin ook niet-integers opgeslagen kunnen worden.

Complexe datatypen:

MODE/RECORD — met behulp van MODE / ENDMODE instruktie wordt de vorm van een record gedefinieerd, met hierin meerdere elementaire variabelen, welke allemaal een eigen veldnaam krijgen. Met RECORD kunnen dan instantiaties van dit type gedeclareerd worden.

ARRAY — een rij van dezelfde elementen, welke met een integer geïndexeerd wordt. De lengte van het array hoeft geen 'compile-time' constante te zijn, en meerdimensionale array's zijn eveneens mogelijk.

FLEXIBLE ARRAY — een array waarvan de lengte niet geïnitialiseerd wordt. Dergelijke arrays worden gevuld door het toepassen van transformaties (TAKES-rules) op andere array's, zodat lengte afhankelijk is van, en pas vastgesteld kan worden na de eigenlijke transformatie.

Al deze complexe datastructuren kunnen 'genest' worden, leidend tot ingewikkelde en krachtige constructies. Onderstaand voorbeeld geeft een idee van de mogelijkheden.

```
MODE score: "deelname resultaten van een student"
  INTEGER inschr_no: "collegekaartnummer"
  INTEGER prakt_cijfer: "cijfer voor praktikum"
  INTEGER tent_cijfer: "cijfer voor tentamen"
ENDMODE

INTEGER aantal_studenten: "aantal studenten waarvoor resultaten"

ARRAY [aantal_studenten] OF RECORD score
    score_tabel: "tabel met deelname resultaten"

FLEXIBLE ARRAY OF INTEGER geslaagd: "de nummers van geslaagden"
```

```
RULE zoek_geslaagden: "maak tabel met geslaagden"
geslaagd TAKES FOR i
      FROM 1 TO length (score_tabel)
      WHERE score_tabel[i].prakt_cijfer > 4 AND
        25* score_tabel[i].prakt_cijfer +
        75* score_tabel[i].tent_cijfer > 549
      (: score_tabel[i].inschr_no:)
```

Het voorbeeld beschrijft de declaratie van een record "score", waarin individuele deelnameresultaten kunnen worden vastgelegd. Er is bovendien een array van dit type record, score_tabel, zodat een tabel met studentresultaten kan worden opgebouwd. We zullen aannemen dat er elders vragen gedefinieerd zijn om deze tabel te vullen. Zodra dit nodig is, probeert het inferentiesysteem eerst de waarde van de variabele "aantal_studenten" vast te stellen, zodat de tabel geïnitialiseerd kan worden. Vervolgens zal het inferentiesysteem de gebruiker vragen ieder van de tabelregels in te voeren.
Een student is geslaagd indien hij tenminste een 4 heeft behaald voor het praktikum, en bovendien zijn gemiddelde cijfer voor het praktikum (25%) en het tentamen (75%) tenminste een 5.5 bedraagt. We willen een overzicht verkrijgen van de collegekaart-nummers van de geslaagde studenten, maar omdat nog niet bekend is hoeveel dit er zullen zijn, gebruiken we een flexible array. In de regel in het voorbeeld wordt dit array (geslaagden) dan gevuld via een tellertje i, dat langs alle regels in de score_tabel loopt, voor iedere regel de test uitvoert en zonodig het collegekaartnummer aan het flexible array geslaagden toevoegt. Op deze wijze gebruikt, krijgt een flexible array het karakter van de uitkomst van een query op een database.

2. Domeinafbakening

Het ligt voor de hand, dat een complex systeem als Envisage een toetssteen van enige omvang verdient om ervaringen boven het triviale niveau uit te tillen. Dit hoofdstuk beschrijft een probleemdomein dat als uitgangspunt heeft gediend voor de ontwikkeling van een proefsysteem.

Kortingsregelingen-probleemdomein

Reizen met het openbaar vervoer in het buitenland kan een kostbare aangelegenheid worden als niet zorgvuldig wordt overwogen welke kortingsregeling voor toeristen van toepassing kan zijn.
Om een keuze uit de diverse regelingen mogelijk te maken, geeft de Nederlandse Spoorwegen in een folder [2] een overzicht van de belangrijkste prijsvoordelen in zo'n vijftien europese landen. Veel van deze regelingen zijn eenvoudig te doorzien, bijvoorbeeld:

Denemarken
Groepsreduktie voor kleine groepen

Reduktie van 20% op de volle enkele reisprijs of retourprijs. Deelnemers dienen gemeenschappelijk en in dezelfde klasse te reizen.

Minimaal 3 personen (volwassenen of kinderen). Bijvoorbeeld 1 volwassene en 2 kinderen. Maximaal 9 personen. Kinderen 4 t/m 11 jaar betalen de helft van de gereduceerde prijs.

Bovendien is dan ook vaak het aantal regelingen beperkt. Bijvoorbeeld, Denemarken: 2, Griekenland: 1. Enkele landen kennen echter een veelheid van, deels elkaar overlappende regelingen met ingewikkelde condities. Bijvoorbeeld, de kilometerbank in Oostenrijk.

Kilometerbank

Aankoop van trein km en gros! 2.000, 5.000 of 10.000 km in de 2e kl. Indien men in de 1e klas reist wordt het aantal km met 1,5 vermenigvuldigd. De kilometerbank voor 2.000 km is 6 maanden geldig; die voor 5.000 en 10.000 km 1 jaar. Minimum af te leggen afstand: 71 km enkele reis.
Maximaal kunnen 6 personen op de kilometerbank reizen. Voor kinderen van 6 t/m 14 jaar wordt de helft van het aantal km gerekend.
De kilometerbank is verkrijgbaar bij de stations en de reisbureaus in Oostenrijk.

Voor een reiziger, die deel uitmaakt van een 'Oostenrijk'-groep van bijvoorbeeld 7 personen waaronder drie kinderen van resp. 2, 4 en 5

jaar, één jongere van 21 jaar en iemand van 71 jaar die 1e klas wil reizen, betekent dit puzzelen en rekenen om zo gunstig mogelijk te kunnen reizen met z'n groep.
Desgevraagd wordt het voor de lokettist(e) ook een puzzelprobleem, hetgeen niet direkt leidt tot een snelle en bevredigende loketafhandeling.

Een expertsysteem met kennis van al deze regelingen zou, door middel van een dialoog, de lokettist(e) kunnen assisteren bij het bepalen van de gunstigste kortingsregeling.
Van het systeem mag worden verwacht dat het:

1. achter het loket kan worden gebruikt middels efficiënt verlopende dialogen;
2. de regelingen adviseert waarvoor de reisgroep in aanmerking komt, maar niet de uiteindelijke tariefbepaling (daar is een ander systeem voor);
3. kennis heeft van de regelingen met een toeristisch karakter in een aantal west-europese landen;
4. over meerdere landen tijdens een consultatie advies kan geven;
5. goed te onderhouden is, zodat regelingen en zelfs landen, zonder al te veel moeite toegevoegd, gewijzigd of verwijderd kunnen worden.

De beschikbaarheid van dokumentatie, de duidelijke afbakening van het probleemgebied en de beperktheid van de complexiteit, maken dit domein tot een bruikbare toetssteen voor ervaringen met Envisage.

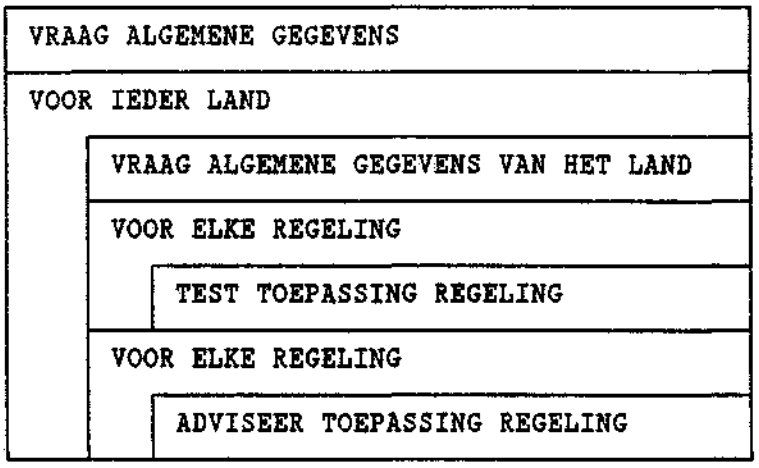
3. Het proefsysteem

Dit hoofdstuk beschrijft een fasegewijze aanpak van het probleem dat in hoofdstuk 2 is afgebakend en beschreven. Het gaat in op het functioneel ontwerp (een resultaat van conceptualisatie), het technisch ontwerp (het gevolg van formalisatie) en de wijze waarop de oplossing is gerealiseerd (implementatie).

3.1 Functioneel ontwerp

Een gebruiker ziet van het systeem voornamelijk twee aspekten, te weten, de dialoog en de adviezen. De dialoog verloopt volgens een vast patroon, dat in een Nassi-Schneidermann diagram alsvolgt kan worden

weergegeven:

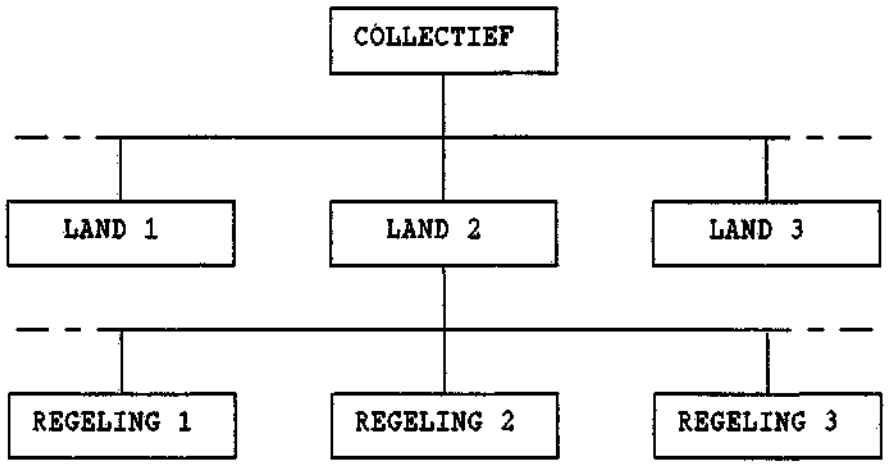


Allereerst vraagt het systeem algemene gegevens, welke in principe voor ieder land relevant kunnen zijn. Dit betreft dan bijvoorbeeld het aantal personen, de leeftijden van de personen, en de landen waarin gereisd zal worden. Vervolgens wordt voor ieder land waarin gereisd zal worden, de gebruiker diepgaander geconsulteerd. De algemene gegevens voor het gekozen land komen aan de orde (indien van toepassing). Dit zijn gegevens die in principe voor alle of de meeste regelingen binnen dat land relevant zijn. Op deze wijze wordt voorkomen dat vragen dubbel gesteld worden. Vervolgens wordt voor elke regeling de toepasbaarheid getest, door te redeneren en door eventueel nieuwe vragen te stellen. Als van alle regelingen vaststaat, of de client ervoor in aanmerking komt, worden de bepalingen van de toepasbare regelingen op het beeldscherm getoond.
De adviezen worden vrij compleet gepresenteerd. De gedachte hierachter is dat indien er meerdere regelingen in aanmerking komen, de cliënt voldoende informatie moet hebben, om de regeling welke hem het meest bevalt te kunnen uitkiezen.

3.2 Technisch ontwerp

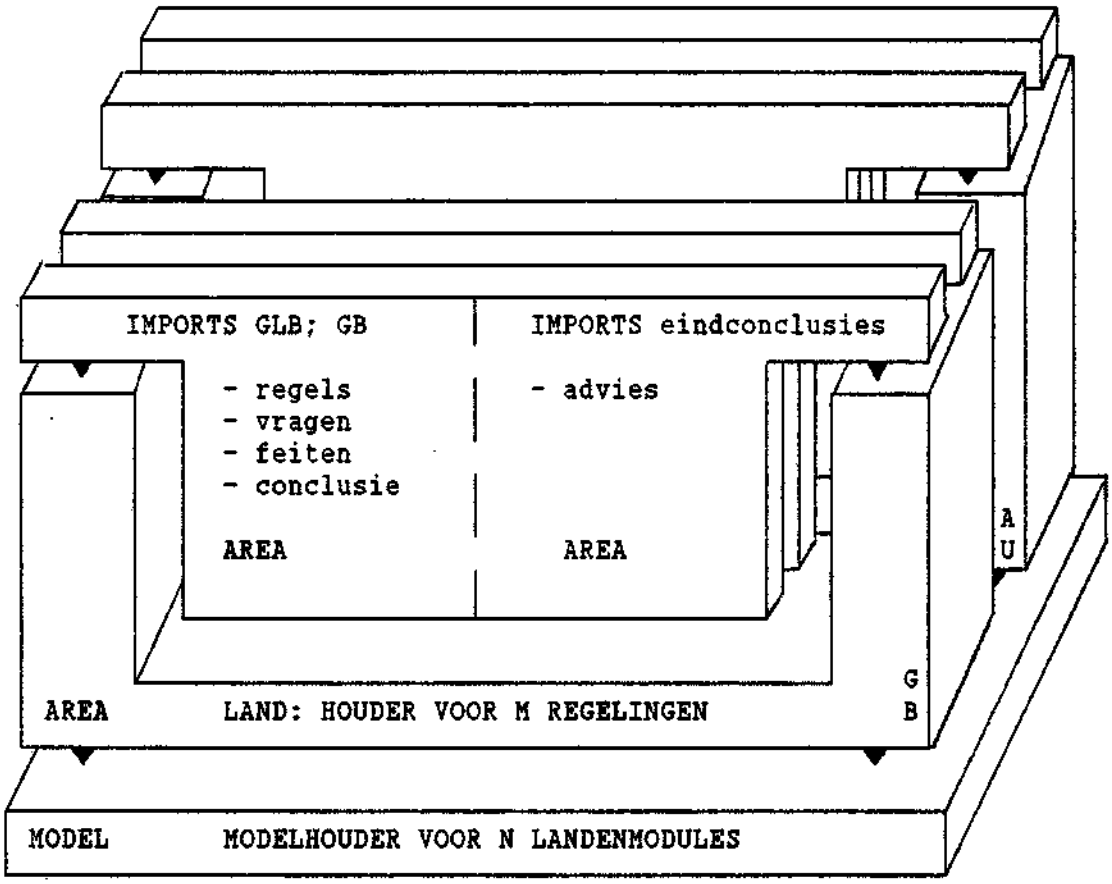
De eis dat het systeem gemakkelijk onderhouden moet kunnen worden, en de wijze waarop structuur van de dialoog is vastgelegd in het functio-

neel ontwerp, hebben tot gevolg dat indeling van het systeem in een hiërarchie van subsystemen voor de hand ligt. We kunnen dit als volgt weergeven:

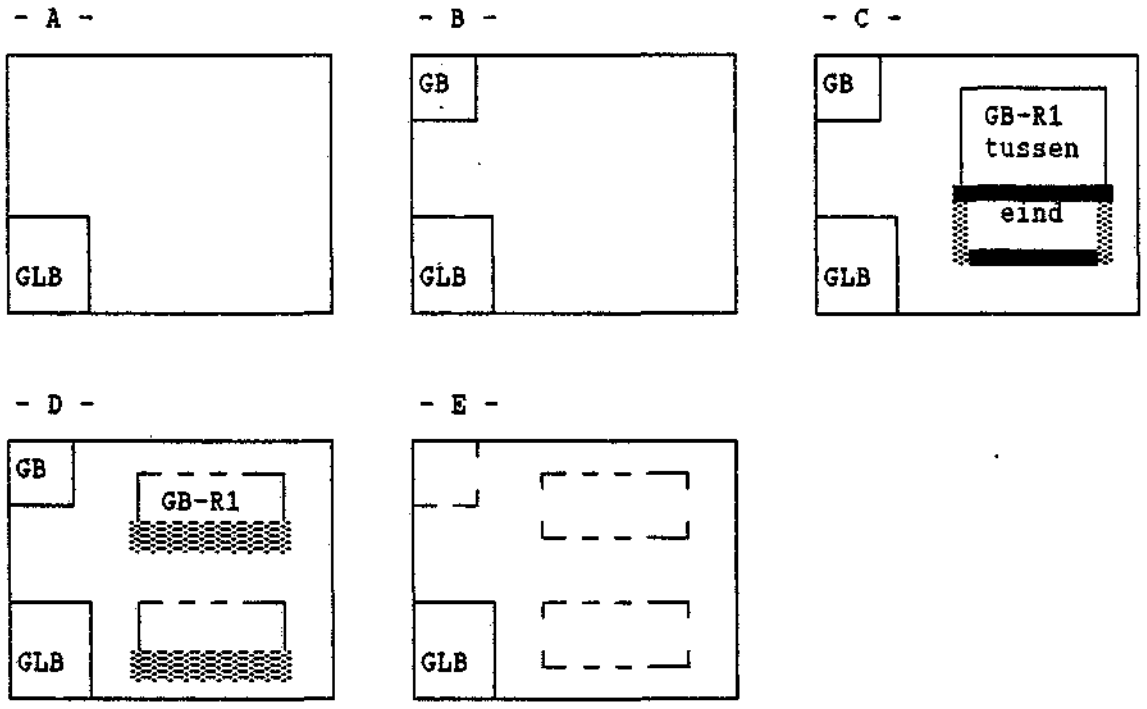


De top van de boom is aangegeven met 'collectief'. Hier worden de algemene gegevens (global data) gevraagd en opgeslagen in een record met de naam GLB. Het onderdeel 'collectief' is op modelniveau gecodeerd, en dus niet in een expliciete area ondergebracht. Ieder land is wel in een aparte area ondergebracht. Deze vormen het tweede niveau binnen de boomstructuur. In deze area's kunnen weer records aanwezig zijn waarin algemene gegevens voor dat land opgeslagen worden. Deze records hebben namen als GB (Groot Brittanië), I (Italië) e.d.
Op het derde hiërarchische niveau vinden we de regelingen. In het Nassi-Schneidermann diagram is te zien, dat iedere regeling als het ware tweemaal bezocht wordt, te weten eerst om te bepalen of de regeling van toepassing is, en daarna om een advies met betrekking tot die regeling uit te brengen. Daarom zijn voor iedere regeling twee area's nodig, bijvoorbeeld gb_rl_test en gb_rl_advise voor de eerste regeling van Groot-Brittanië.
De areas op verschillende niveaus communiceren onderling door middel van het importeren van variabelen. De tweede niveau areas importeren

GLB, de algemene gegevens. De area's op het derde niveau (de test of een regeling toepasbaar is), importeren behalve GLB, ook de algemene gegevens voor dat specifieke land. De area's om over een regeling te adviseren, importeren van de bijbehorende testarea een variabele met daarin 'het advies'.
We kunnen de architectuur van het systeem als volgt uitbeelden:



Er kunnen dus gemakkelijk regelingen toegevoegd en weggenomen worden, want er zijn in feite maar twee aansluitpunten. Het toevoegen en verwijderen van landen is al evenmin een probleem, want de bij dat land behorende area's staan helemaal los van de rest.
Om een en ander nog te verduidelijken, zullen we nagaan hoe de variabelen in de werkruimte van he systeem zich ontwikkelen.
In fase A is de consultatie net begonnen. In het record GLB zijn de algemene gegevens verzameld. In fase B wordt aangevangen met de consultatie voor een bepaald land. De bijbehorende area wordt aangeroepen, en de algemene gegevens voor dat land worden ook weer gevraagd en in een record opgeslagen.



Vervolgens worden in fase C de regelingen stuk voor stuk getest door de bijbehorende test-area aan te roepen. Iedere area kan tussenconclusies hebben, maar heeft in ieder geval een eindconclusie. Deze eindconclusie wordt bewaard en later geraadpleegd door de adviserende area. In fase D zijn alle regelingen getest, en is voor iedere regeling nog een variabele beschikbaar met de eindconclusie. Vervolgens worden in fase E de adviezen gegeven, en is de consultatie voor dat land klaar. Alle variabelen welke tijdens de consultatie voor dat land gegenereerd zijn, blijven latent in de computer achter en worden verder niet meer geraadpleegd. De consultatie is nu beëindigd, of wordt voortgezet voor een ander land, in welk geval het proces weer opnieuw begint.

3.3. Implementatie

Om bovenstaande architectuur in praktijk te brengen, werd een compleet model, met echter maar één land, te weten Groot-Brittannië, geïmplementeerd. Aan de hand van dit model kon de opzet van de modelhouder getoetst en uitgewerkt worden. Vervolgens werden uit een kopie van dit model alle area's, welke betrekking hadden op Groot-Brittannië, verwijderd, zodat alleen de modelhouder overbleef. Kopieën van deze modelhouder vormden de basis om de overige landen te coderen en te testen. Hierbij werden zoveel mogelijk de conventies met betrekking tot naamge-

ving en indeling die in het model van Groot-Brittanië besloten lagen, aangehouden. Alle op deze wijze ontwikkelde land-modules, konden eenvoudig worden samengevoegd door het kopiëren van alle area's naar één bestand.
Toegevoegde CONSIDER's verzorgden de aansluiting tussen de verschillende delen en diverse 'verfraaiingen' in deze eind-redactie-fase resulteerden in een bruikbaar eindprodukt.

4. Evaluatie

In dit hoofdstuk komen de belangrijkste ervaringen met Envisage aan de orde. Hierbij moet aangetekend worden dat de kritiek die wij zullen formuleren waarschijnlijk sterk beïnvloed is door de structuur van het NS-project dat wij met behulp van Envisage hebben proberen te realiseren. Bij een ander probleem zouden sommige punten van kritiek misschien niet zo zwaar wegen, en wellicht ook andere punten naar voren komen. De evaluatie spitst zich toe op achtereenvolgens de syntax, het procedureel karakter en de datatypen van Envisage.

Syntax

De taaldefinitie van Envisage zoals vastgelegd in het reference manual [3], geeft duidelijk uitsluitsel over de toegelaten constructies. De meeste keywords, en de grotere verbanden waarin ze gebruikt moeten worden, appeleren al na korte tijd aan het programmeursgevoel voor logica en structuur. Een belangrijk punt van ergernis bij programmeren is echter de grote hoeveelheid 'commentaarstrings' die bij ieder object (zoals variabelen, regels) vermeld moeten worden. Naast de gewone objectnaam (b.v. rode_drank) moet dus iets als "het drankje heeft een rode kleur" getypt worden. Zoals al uit het voorbeeld blijkt, krijgt de string vaak een wat redundant karakter. Sommige strings worden zo nu en dan wel gehanteerd in de dialoog met de gebruiker, maar gezien het gekunstelde karakter van dergelijke zinssneden zal de programmeur dit over het algemeen vermijden. Het zou dan ook aanbeveling verdienen om de commentaarstring facultatief te stellen.

Procedureel vs. declaratief

Een expert-shell maakt het mogelijk om kennisstructuren als het ware te declareren, zodat het procedurele aspect van het redeneren niet gespecificeerd behoeft te worden. De shell zoekt zelf redeneerpaden, en tracht bewijs te verzamelen voor bepaalde hypothesen.
Toch ontkomt ook de programmeur van een expert-shell er niet aan, om procedurele elementen te specificeren. Dit betreft dan vooral de redeneerdoelen en het verloop van de dialoog.
En Envisage is het procedurele aspect van het programmeren belichaamd in de acties. Alle acties op modelniveau of binnen een area worden sequentieel uitgevoerd, mits aan de PROVIDED-voorwaarde voor die actie is voldaan. De mogelijkheden (in de zin van selectie en herhaling) om een structuur in de dialoog aan te brengen zijn beperkt. Dit komt scherp naar voren in het NS-model. Voor alle regelingen zijn twee areas nodig, één om te redeneren en om vragen te stellen, en één om het advies te geven. Deze constructie is noodzakelijk indien we per land vragen van de adviezen willen scheiden. Indien Envisage meer mogelijkheden bood om de volgorde van actions binnen areas te besturen, zou wellicht een hoop overhead (areas, imports) vermeden kunnen worden.
Ook komt de behoefte aan meer procedurele statements naar voren bij in en uitvoer van gestructureerde data-items, in het bijzonder bij arrays. Er is voorzien in een 'for' declaratie, welke 'loopjes' kan genereren. Deze constructie is echter niet algemeen bruikbaar.
Wellicht zou het specifieke expert-shell karakter van Envisage door dergelijke toevoegingen verloren gaan, maar voor het bouwen van krachtige decision support systemen zouden ze toch van grote waarde zijn.

Datatypen

Het assortiment elementaire datatypen (string, integer, object (real) en assertion (boolean)) omvat helaas niet het 'enumerated' type. Dit type beschrijft een variabele die in twee of meer statussen kan verkeren, welke alle met een bepaalde term aangeduid worden. Bijvoorbeeld: geslacht = man, vrouw; boek = pocket, paperback, gebonden. Indien we dergelijke zaken nu in de redenatie op willen nemen, moeten ze gecodeerd worden naar verschillende waarden van een integer, en vervolgens menu's en scales gebruiken om ze te kunnen in en uit voeren. Het ontbreken hiervan wordt wel als gemis ervaren.

Zoals hierboven al is vermeld, kunnen records en arrays worden gedeclareerd. Vooral ten aanzien van arrays moet van het nut hiervan geen al te grote voorstellingen gemaakt worden, want de enige, voor de hand liggende manier waarop een array gevuld kan worden is in de dialoog via het toetsenbord. Het is duidelijk dat men van de gebruiker niet kan verlangen om tientallen of honderden soortgelijke gegevens tijdens een dialoog voor eenmalig gebruik in te voeren.
Voor wat betreft de mogelijkheid tot het creëren van records kunnen we soortgelijke opmerkingen maken. Een recordtype beschrijft de attributen van een entiteittype, en een entiteittype als 'ENVISAGE-object' heeft vooral zin als er meerdere occurences van zijn. Dan komen we al gauw weer in array-sferen terecht. zodat de hierboven gemaakte opmerkingen in versterkte mate gelden.
Toch is het duidelijk dat veel gegevens, vormen aannemen die een array- of recordbeschrijving vereisen en het behoeft geen toelichting dat deze gegevens in een consultatie relevant kunnen zijn. Envisage zou dan ook sterk aan waarde winnen indien behalve uit een dialoog, gegevens verkregen zouden kunnen worden middels queries op aanwezige databases. In de huidige versie is hierin niet voorzien, hoewel er genoeg aanknopingspunten in de taalconstructies te vinden zijn. Een FLEXIBLE ARRAY OF RECORD ... is bij uitstek geschikt om de resultaten van b.v. een SQL-query te accommoderen. Een regel om een dergelijk array te vullen zou dan bijvoorbeeld als volgt kunnen luiden:

```
MODE    supplier:   "gegevens van een leverancier"
  STRING    snum:   "supplier identificatie"
  STRING   sname:   "naam supplier"
  INTEGER status:   "status supplier"
  STRING    city:   "stad supplier"
ENDMODE

FLEXIBLE ARRAY OF RECORD suppliers
    london_suppliers: "suppliers in london"

RULE query_suppliers: "voer queries uit op database"
    london_suppliers QUERIES
        SELECT  *
          FROM  S
          WHERE CITY = 'LONDON'
```

Hoewel de portabiliteit van een Envisage-versie die dergelijke constructies toestaat, minder is, zou zij zonder twijfel de gebruiksmogelijkheden aanzienlijk vergroten. Bovendien moet implementatie van de interface naar de database niet zo'n groot probleem zijn.

Tenslotte

Uit de beschrijving in hoofdstuk 2 kan worden geconcludeerd dat Envisage een geavanceerd, op regels gebaseerd produkt is. Het kent een veelheid aan begrippen om kennis te representeren, het redeneerproces te sturen, evenals de dialoog. In deze veelheid schuilt ook een gevaar. Bij een goed gereedschap is een evenwicht bereikt tussen het gewenste aantal vrijheidsgraden (de inzetbaarheid) enerzijds en de conceptuele eenvoud anderzijds.
Envisage kent, behoudens de kritiek in hoofdstuk 4, vele mogelijkheden om formalisatie- en implementatieproblemen te trotseren (de naam zegt het al). Goed omgaan met Envisage is echter niet eenvoudig en het bordje van knowledge engineer is al zo vol.

Literatuur

[1] Hayes-Roth, F. et. al. 'Building Expert Systems', Addison-Wesley, Reading, 1983.

[2] 'Voordelige treintarieven in europese landen', NS uitgave juni 1986, Rd/Mr 1.1.

[3] Envisage Reference Manual, ENVø2, Issue 3.0, Systems Designers, June 1986.

1986-1	Peter Nijkamp	New Technology and Regional Development
1986-2	Floor Brouwer Peter Nijkamp	Aspects and Application of an Integrated Environmental Model with a Satellite Design (E 85/4)
1986-3	Peter Nijkamp	25 Years of Regional Science: Retrospect and Prospect
1986-4	Peter Nijkamp	Information Centre Policy in a Spatial Perspective
1986-5	Peter Nijkamp	Structural Dynamics in Cities
1986-6	Peter Nijkamp Jacques Poot	Dynamics of Generalised Spatial Interaction Models
1986-7	Henk Folmer Peter Nijkamp	Methodological Aspects of Impact Analysis of Regional Economic Policy
1986-8	Floor Brouwer Peter Nijkamp	Mixed Qualitative Calculus as a Tool in Policy Modeling
1986-9	Han Dieperink Peter Nijkamp	Spatial Dispersion of Industrial Innovation: a Case Study for the Netherlands
1986-10	Peter Nijkamp Aura Reggiani	A Synthesis between Macro and Micro Models in Spatial Interaction Analysis, with Special Reference to Dynamics
1986-11	E.R.K. Spoor	De fundamenten van LINC
1986-12	V. Kouwenhoven A. Twijnstra	Overheidsbetrekkingen in de strategie en organisatie van ondernemingen
1986-13	F.C. Palm E. Vogelvang	A short run econometric analysis of the international coffee market
1986-14	M. Wortel A. Twijnstra	Flexibele Pensioenering
1986-15	A. de Grip	Causes of Labour Market Imperfections in the Dutch Construction Industry
1986-16	F.C. Palm C.C.A. Winder	The Stochastic life cycle consumption model: theoretical results and empirical evidence
1986-17	Guus Holtgrefe	DSS for Strategic Planning Purposes: a Future Source of Management Suspicion and Disappointment?
1986-18	H. Visser H.G. Eijgenhuijsen J. Koelewijn	The financing of industry in The Netherlands
1986-19	T. Wolters	Onderhandeling en bemiddeling in het beroepsgoederenvervoer over de weg
1986-20	S.C.W. Eijffinger J.W. in 't Veld	De relatieve posities van de EMS-valuta's
1986-21	E.R.K. Spoor	Knowledge Eliciting and Modelling using the entity Relationship Approach
1986-22	J.T.C. Kool A.H.Q.M. Merkies	On the Integration of Multi-step Prediction and Model Selection for Stationary Time Series
1986-23	W. van Lierop L. Braat	Multi-Objective Modelling of Economic-Ecological Interactions and Conflicts
1986-24	R.M. Buitelaar J.P. de Groot H.J.W. Wijland	Agrarian Transformation and the rural labour market: the case of Nicaragua
1986-25	E.R.K. Spoor & S.J.L. Kramer	Revaluatie van het E-R model
1986-26	Herman J. Bierens	Armax model specification testing, with an application to unemployment in the Netherlands
1986-27	Jan Rouwendal Piet Rietveld	Search and Mobility in a Housing Market with Limited Supply
1986-28	W. Keizer	The concept of 'consumer' sovereignty Exposition, critique and defense.
1986-29	Max Spoor	Finance in a socialist transition: the case of the republic of Viet Nam (1955-1964)
1986-30	L.J.J. van Eekelen	De ontwikkeling van het solvabiliteitstoezicht van de Nederlandsche Bank
1986-31	F.A. Roozen	Een verkenning naar invloeden van flexibele produktie automatisering op het management informatie systeem
1986-32	H.J. Bierens	Model-free asymptotically best forecasting of stationary economic time series
1986-33	R. Huiskamp	Large corporations, industry bargaining structures and national industrial relations: a comparitive and organisational approach
1986-34	W.J.B. Smits	Directe buitenlandse investeringen: invloed op export- en importwaarde; een cross-section analyse voor 30 ontwikkelingslanden
1986-35	P. Nijkamp	Informatics or oracles in regional planning
1986-36	H. Blommestein P. Nijkamp	Adoption and diffusion of innovations and evolution of spatial systems