BEELDSCHERMKARTOGRAFIE TEN BEHOEVE VAN MULTI-BRON INTERNET GIS

INGRID ALKEMADE

TU Delft

fotomanipulatie

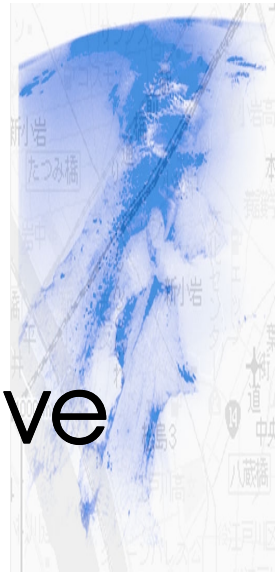
BEELDSCHERMKARTOGRAFIE TEN BEHOEVE VAN MULTI-BRON INTERNET GIS

AFSTUDEERSCRIPTIE INGRID ALKEMADE
DELFT, DECEMBER 2000

AFSTUDEERHOOGLERAAR PROF.DR.IR. PETER VAN OOSTEROM
AFSTUDEERBEGELEIDER DRS. THEO TIJSSEN

FACULTEIT CIVIELE TECHNIEK EN GEOWETENSCHAPPEN
AFDELING GEODESIE
SECTIE GIS TECHNOLOGIE-

TU Delft



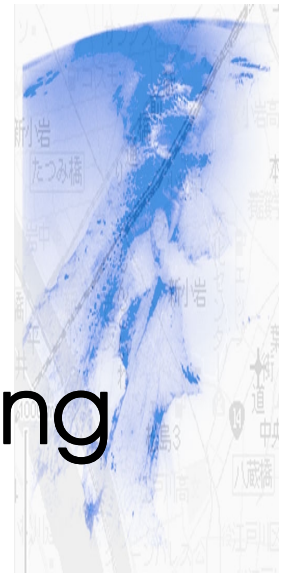
Inhoudsopgave

Voorwoord

Deze scriptie, *Beeldschermkartografie ten behoeve van een multi-bron internet GIS*, is het resultaat van het afstudeeronderzoek als sluitende opdracht van de studie Geodesie aan de Technische Universiteit Delft. Het afstudeeronderzoek heeft als doel aan te tonen dat een student in staat is individueel kan werken, de gestelde taken kan overzien, een onderzoek op juiste wijze kan verrichten en de resultaten van dit onderzoek kan rapporteren en presenteren.
Het onderzoek is uitgevoerd bij de afdeling Geodesie (TU Delft) bij de sectie GIS-technologie. De sectie GIS-technologie verzorgt onderwijs, verricht onderzoek en verleent diensten op het gebied van de technologische aspecten van geografische informatiesystemen. Geografische informatiesystemen (GIS) spelen hierbij een centrale rol, te meer omdat (mede door de snelle opkomst van de mogelijkheden van het internet) het gebruik van GIS alleen maar toeneemt. Complexe problemen doen zich voor met betrekking tot de integratie van geografische bestanden, de kwaliteitsbeschrijving van geografische gegevens en de visualisatie en bevraging van geografische gegevens. De GIS technologie slaat een brug tussen de technieken voor inwinning van de ruimtelijke gegevens en het gebruik van geografische informatie, en is als zodanig de verbindende factor van de verschillende secties van de afdeling Geodesie.

Veel dank is verschuldigd aan de vele mensen die zich hebben ingespannen om dit onderzoek mogelijk te maken. In de eerste plaats natuurlijk de beide begeleiders Theo Tijssen en Peter van Oosterom voor hun inspanningen en nuttige adviezen en opmerkingen. Daarnaast bedank ik Frank Tuijnman van PGS voor het beschikbaar stellen van de Magma/Lava software, de gemeente Tilburg voor het beschikbaar stellen van hun datasets en de waardevolle adviezen die zij mij gaven vanuit een praktische omgeving.

Delft, 6 december 2000
Ingrid Alkemade



Samenvatting

Door de snelle opkomst van de mogelijkheden van het internet, is het gebruik van Geografische Informatiesystemen (GIS) toegenomen. Het beschikbaar stellen van geografische gegevens via internet kan op twee manieren gebeuren: de eerste manier is dat de server een kaart maakt van de gevraagde gegevens en deze in de vorm van een GIF of JPEG (rasterbeeld) naar de gebruiker stuurt (server side), de tweede manier is om de gegevens zelf naar de gebruiker te sturen, zodat deze zelf een kaart naar wens kan produceren (client side). De GIS-functionaliteit ligt in het eerste geval bij de server, in het tweede geval bij de client. De meeste geografisch georiënteerde sites zijn echter atlassen en geen GIS producten, omdat de mogelijkheden van analyse, bevraging en weergave beperkt zijn.

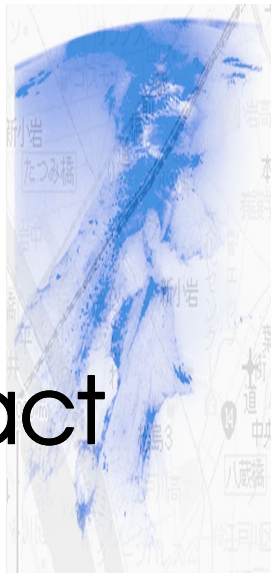
Een nieuwe ontwikkeling in internet GIS is dat er gebruik wordt gemaakt van gedistribueerde bronnen: de gebruiker kan een keus maken uit gegevens van verschillende databronnen. De huidige ontwikkeling bij de levering van geografische bestanden is dat naast de data zelf ook de zogenaamde meta-informatie als formaat, projectie, attribuutdefinitie, actualiteit en nauwkeurigheid (kwaliteit van de gegevens), wordt bijgeleverd. Deze meta-informatie stelt de gebruiker in staat te bepalen in hoeverre de data geschikt zijn voor gebruik binnen een GIS. Het Open GIS consortium ontwikkelt de standaards voor uitwisseling van geografische informatie.

Bij de visualisatie van de gegevens kunnen in een multi-bron situatie echter gemakkelijk conflicten ontstaan, omdat er door de verschillende bronnen geen voorschriften worden gegeven over de grafische representatie van de gegevens. Het is daarom nuttig een onderscheid te maken tussen het Digitaal Landschaps Model (DLM), waarin de gegevens zelf worden beschreven, en het Digitaal Kartografisch Model (DKM) dat een beschrijving geeft van de grafische weergave van de geografische data. Voor een internet GIS is het wenselijk dat de bron naast het DLM ook een voorkeurs DKM opstelt. Bij een combinatie van gegevens uit verschillende bronnen wordt er gebruik gemaakt van een expertsysteem dat met behulp van kartografische regels bepaalt wat de optimale presentatie van de gegevens is, gegeven de taak en de wensen van de gebruiker. Dit Kartografisch Expertsysteem (KES) is erop gericht de gebruiker te begeleiden en te adviseren in het maken van een kartografisch verantwoord product.

Het KES gebruikt kartografische regels in de vorm van een IF *[toepasbare voorwaarde]* THEN *[aanbeveling]* waarbij de voorwaarden de feiten en procedurele kennis omvatten en de aanbeveling een taak of verwijzing naar een volgende regel is. Het is belangrijk te onderkennen dat de eerste stappen van een expertsysteem vooral van toepassing zijn op de minder complexe onderdelen van de kartografische regels. Het idee van een intelligent en gebruikersvriendelijk GIS impliceert dat het niet alleen een verzameling van een aantal krachtige instrumenten is die gebruikt kunnen worden voor het weergeven en analyseren van gegevens, maar dat het de gebruiker ook kan adviseren in het weergeven van de gegevens.

In de ontwikkeling van een kartografisch expertsysteem zijn er twee belangrijke onderdelen. Het *eerste onderdeel* is om bestaande kartografische kennis om te zetten in regel-gebaseerde kennis, dus hoe de kennis ingevoerd wordt. Er zijn een aantal aandachtspunten in het grafische model te onderscheiden die in regels gevat zouden kunnen worden. Deze aandachtspunten zijn: schaal en schaaldomein, generalisatie, semantiek, plaatsing van tekst, belang van thema's, visualisatie en voorschriften van de bron. Het *tweede onderdeel* is een leek begeleiden in het maken van een kaart, dus het gebruiken van de regels die in door het eerste onderdeel zijn opgesteld. Aan de hand van de genoemde aandachtspunten zullen regels opgesteld moeten worden die het automatiseren mogelijk maken.

Hoewel sommige regels op het eerste gezicht eenvoudig lijken, kunnen deze praktisch vrijwel onuitvoerbaar zijn, omdat het beslisproces wat er aan te pas komt te complex is om op een eenvoudige manier te implementeren in een geautomatiseerd systeem. Het gaat hierbij vooral om regels met betrekking tot generalisatie en semantiek: hoe kun je een formele beschrijving geven van een weg of een gebouw, hoe kun je menselijke interpretatie en handelen bij generalisatie in regels vatten?
Het geïmplementeerde voorbeeld van één van de minder complexe regels (het oplossen van kleurconflicten), laat zien dat het mogelijk is om met behulp van visualisatievoorschriften te komen tot een juiste visualisatie. De gebruiker speelt hierbij een actieve rol door binnen de legenda aan te geven wat zijn prioriteiten zijn en zo impliciet het doel van de visualisatie aangeven.



Abstract

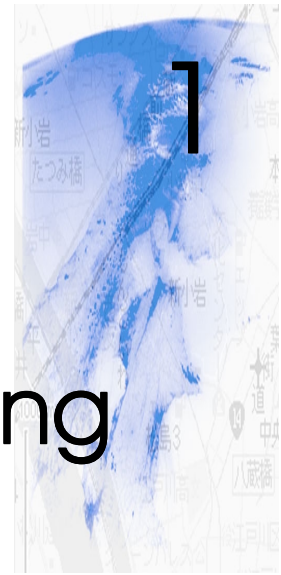
The quick rise of the possibilities of the internet has caused an increased use of Geographic Information Systems (GIS). Internet provides a platform to distribute geographical data. However, most of the current geographic sites are atlases and not GIS products, because they are short of possibilities for analysis, query and visualisation. A new development in internet GIS is the use of distributed sources: the user can make his choice between diverse data-sources. Geographic data is currently delivered as the data itself and meta-data like format, projection, attribute-definition, actuality and accuracy (quality of the data). The meta-information enables the user to decide whether the data is suitable for the use in a GIS. The OpenGIS Consortium (OGC) is developing standards for interoperability of geographic data.
In the case when multiple sources are used, the visualisation of the datasets can cause conflicts for there is no standard or direction for the graphical presentation of the data. Therefore, it's useful to make a distinction between the Digital Landscape Model (DLM) where the data itself is present, and the Digital Cartographic Model (DCM, in Dutch DKM) where the graphical presentation is described. For the use of an internet GIS, it is important that not only the DLM, but also the DKM is provided by the data source. When a combination of several DLM's is made, the use of an expert system is needed, which uses cartographic rules to determine the optimal presentation of this set of data, given the user's task and preferences. This Cartographic Expert System (CES, in Dutch KES) is aimed to accompany and advise the user in producing a cartographic decent product.

A CES uses cartographic rules in the form of an IF [applicable condition] THEN [recommendation] where the applicable conditions are facts and procedural knowledge and a recommendation is a task or a reference to another rule. It's important to recognise that the first steps that lead to an expert system, set in this MsC thesis, are only for less complex applications.
The idea of an intelligent an user-friendly GIS implicates that it is not only a collection of some powerful tools that can be used for presenting and analysing data, but it also makes recommendations to the user for presenting the data.

The development of a CES has two major parts. The *first part* is to transform cartographic knowledge into rule-based knowledge, this means how the knowledge is put into the system. Some points of attention can be distinguished in the graphical model that can be

seized into rules. Those points of attention are scale and scale domain, generalisation, semantics, text placement, importance of themes, visualisation and the recommendations of the data-source. The second part is to accompany the laymen in producing a map. On the basis of the points of attention, rules will be drawn up that make automation possible.

Although some rules may look simple at first sight, they can turn out to be very difficult. The process of deciding that is involved can be too complex to be implemented in an automised system in a relatively simple way. Especially rules that involve generalisation and semantics can be very complex.
The implemented example of one of the less complex rules (i.e. solving a conflict in colours) shows that it is possible to use recommendations of visualisation to create an optimal presentation of several datasets. The user has an active role by manipulating the legend, thereby setting his priorities and thus indicates the goals in an implicit way.



1 Inleiding

Door de toename van het gebruik van internet en de vele mogelijkheden die het biedt, gebeurt het dat steeds meer mensen via het internet toegang hebben tot geografische informatie. Een Geografisch Informatiesysteem (GIS) is een systeem voor het opslaan, controleren, integreren, manipuleren, analyseren en weergeven van deze geografische informatie. De gegevens zelf kunnen beschreven worden als een Digitaal Landschapsmodel (DLM), de grafische weergave hiervan als een Digitaal Kartografisch Model (DKM) [Van der Schans, 1999].
Het beschikbaar stellen van geografische gegevens via internet kan op twee manieren gebeuren. De eerste manier is dat de server een kaart maakt van de gevraagde gegevens en deze in de vorm van een GIF of JPEG (rasterbeeld) naar de gebruiker stuurt. De tweede manier is om de gegevens zelf (oftewel het DLM) naar de gebruiker te sturen, zodat deze zelf een kaart naar wens kan produceren. De GIS-functionaliteit ligt in het eerste geval bij de server, in het tweede geval bij de client.
De meeste geografisch georiënteerde sites op internet zijn atlassen en geen GIS producten. Het ondervragen van het systeem beperkt zich meestal tot het aan- en uitzetten van verschillende lagen en het vinden van locaties. De beperkingen die de huidige systemen op internet hebben, zijn vaak gericht op het feit dat de gebruikers geen experts zijn op het gebied van kartografie. De meeste GIS-functionaliteit bevindt zich dus aan de server-kant, en beperkt zich tot slechts één server die alle data per site levert. Het resultaat is een voorgeproduceerde kaart die, bij weergave op het beeldscherm, kartografisch in orde is.

Een nieuwe ontwikkeling in internet-GIS is dat er gebruik wordt gemaakt van gegevens van meerdere databases, dus van gedistribueerde databronnen. In de toekomst moet het mogelijk worden geografische gegevens van bijvoorbeeld het Kadaster via internet op te vragen. Deze zogenaamde loketfunctie kan ervoor zorgen dat de informatie toegankelijker wordt terwijl de gegevens bij de bron kunnen blijven. Het gebruik van verschillende bronnen, of gedistribueerde databases, kan echter voor visualisatieconflicten zorgen. Iedere bron of database heeft zijn eigen DKM dat voor deze enkele bron het beste is. Vaak is dit model ook gebaseerd op een heersende conventie, water wordt bijvoorbeeld in de meeste gevallen met blauw aangeduid. Het is echter niet zo dat de combinatie van visualisaties van verschillende bronnen dan ook juist is.
Het gebruik van meerdere databronnen heeft dus gevolgen voor de kartografische regelgeving en het opstellen van een DKM. Het gebruik van een GIS op internet impliceert

dat de kartografische kennis niet altijd aanwezig is. Om toch een goede kaart op het beeldscherm te krijgen en tegelijkertijd de gegevens van deze kaart te kunnen analyseren moeten er algemene kaartontwerpregels worden opgesteld in de vorm van een Kartografisch Expertsysteem (KES). Deze set van regels vormt van de verschillende kartografische modellen van de bronnen een nieuw, geïntegreerd DKM, zodat het KES de gebruiker adviseert en begeleidt tot een kartografisch verantwoord product. De gebruiker is hierbij sturend door zelf aan te geven welke gegevens het belangrijkst zijn om het doel van zijn visualisatie te bereiken.

In deze afstudeeropdracht wordt een begin gemaakt met het opstellen van een KES in de vorm van een set van kartografische regels die van gedistribueerde DLM met een eigen DKM een gezamenlijk DKM kan maken zonder visualisatieconflicten. Enkele van deze kartografische regels worden geïmplementeerd in een bestaande GIS-browser [Magma/Lava] die de mogelijkheid geeft tot uitbreiding. De regels combineren data van verschillende bronnen tot een verantwoorde visualisatie. De nadruk ligt daarbij op begeleiding en advisering. Bij selecties binnen de opgevraagde data moeten de kartografische regels opnieuw toegepast worden en de data opnieuw juist visualiseren. De gebruiker zelf moet altijd de eindbeslissing kunnen nemen, ook als dat een product oplevert die niet binnen de kaartontwerpregels van het DKM vallen.
De GIS browser is een Java-applet, dit heeft als voordeel dat Java-applets platform-onafhankelijk zijn. Het kan dus op iedere computer gebruikt worden en heeft geen apart conversieprogramma nodig. Zeker met het oog op de toekomstige algemene toegang tot geografische informatie op het internet is platform-onafhankelijkheid een belangrijke voorwaarde voor het systeem.

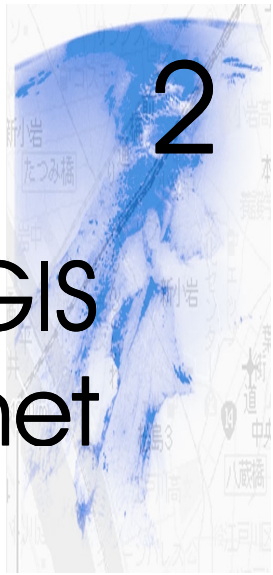
Dit afstudeeronderzoek is een onderdeel van een groter onderzoeksproject, dat niet alleen de kartografische aspecten van een interoperationeel GIS omvat. Er wordt onder andere onderzoek gedaan naar de volgende onderwerpen:

- De huidige gebruikte DBMS is het relationele Oracle 8i DBMS [Herbert, 1999]. Ook gegevens uit andere DBMSsen (verschillende merken, types en versies) moeten verwerkt kunnen worden;
- Magma/Lava gebruikt nu haar eigen protocol om te communiceren met de verschillende magma-servers. Het toepassen van het gestandaardiseerde protocol dat ontwikkeld is door OpenGIS (Web Map Server Specifications, zie ook § 2.3.4) biedt de mogelijkheid om Magma/Lava een meer open structuur te geven, zodat ook met andere servers dan de eigen magma-server gecommuniceerd kan worden;
- Het gebruik van GML en XSLT (zie ook § 2.3.5) voor het verzenden van data respectievelijk voorkeursvisualisatie wordt onderzocht.

Hoewel de verschillende gegevensbronnen op het moment feitelijk nog op één systeem te vinden zijn, wordt er in de opzet van het onderzoek zoveel mogelijk gestreefd naar een systeem dat in essentie een internet GIS met gedistribueerde bronnen is. Fysiek zijn de bronnen dus niet gedistribueerd, maar de huidige opzet biedt een simulatie van het toekomstige internet GIS.

In hoofdstuk 2 zullen basisbegrippen als DLM, DKM en internet GIS nader toegelicht worden. Voor het gebruiken van een internet GIS zijn er een aantal nieuwe ontwikkelingen aan de orde die mogelijk een bijdrage kunnen leveren aan een breed toegankelijke GIS-browser. Deze nieuwe ontwikkelingen worden ook besproken, hoewel deze in dit onderzoek

verder niet gebruikt zullen worden. Hoofdstuk 3 zal ingaan op kartografische expertsystemen, de beperkingen die het internet daaraan stelt en de regels van beeldschermkartografie.
In hoofdstuk 4 wordt de internet-browser (Lava) die in dit onderzoek gebruikt werd beschreven, inclusief de implementatie van de nieuwe regels omtrent de kartografische vormgeving. Ten slotte zullen in hoofdstuk 5 de conclusies en aanbevelingen worden beschreven.



2 Kartografie, GIS en internet

Met de komst van nieuwe digitale technieken zijn een aantal traditionele technieken aan sterke verandering onderhevig. Waren vroeger bijna alle kaarten afgebeeld op papier, tegenwoordig zijn er diverse mogelijkheden om geografische informatie te presenteren. De traditionele kartografie wordt uitgebreid met een nieuwe dimensie: de digitale kaart. Dit brengt met zich mee dat de kartografische theorie niet zondermeer toepasbaar is op geografische datasets die op een beeldscherm gevisualiseerd worden. In dit hoofdstuk wordt ingegaan op de kartografische theorie en de samenhang met digitale technologieën als GIS en internet. In § 2.1 wordt ingegaan op de kartografische theorie en beeldschermkartografie en worden het DLM en DKM beschreven. De samenhang tussen GIS en internet wordt in § 2.2 beschreven. In dezelfde paragraaf worden ook enkele GIS producten op internet vergeleken. § 2.3 gaat in op het gebruik van meerdere bronnen en onderwerpen als standaardisatie, OpenGIS en XML. Tenslotte worden in § 2.4 de onderdelen van een internet GIS beschreven en de plaats die de kartografische theorie daarin inneemt.

2.1 Kartografische theorie

2.1.1 Kartografisch ontwerp

Kartografie is gericht op het zichtbaar maken van ruimtelijke informatie. Een mogelijke definitie van kartografie is [naar van der Schans, 1999]:

> Het geheel van wetenschappelijke, technische en artistieke activiteiten gericht op de vervaardiging en het gebruik van kartografische producten. Het gaat hier niet alleen om kaarten, maar ook om kaartverwante afbeeldingen en bestanden die bedoeld zijn voor visualisatie.

Deze kaarten worden onderscheiden in twee hoofdonderdelen: *topografische kaarten* (die direct zichtbare terreinobjecten weergeven zoals wegen en gebouwen) en *thematische kaarten* (die de spreiding van een bepaald verschijnsel weergeven zoals de spreiding van de bevolkingsdichtheid van een land).
Het ontwerp van een kaart is sterk afhankelijk van het doel van de kaart. Er moeten keuzes gemaakt worden voor de kartografische methode die gebruikt gaat worden (het soort kaart, topografische of thematische kaart), de minimale of maximale schaal die de kaart kan

hebben, de klassen of aggregatieniveaus waarin de gegevens ingedeeld worden en de geschikte meetschaal voor de uit te beelden gegevens [Bertin, 1983].

Voor het afbeelden van ruimtelijke elementen zijn er grafische elementen nodig die een relatie kunnen leggen tussen het teken dat afgebeeld is en de betekenis dat het afgebeelde heeft. De Franse cartograaf Jacques Bertin stelde in 1967 als eerste een aantal variabelen voor [Bertin, 1983][1], die de basis bleken voor alle afbeeldingen van gegevens in kaarten. Deze variabelen zijn gericht op de aspecten op grond waarvan grafische elementen van elkaar kunnen worden onderscheiden. Bertin stelde dat een visueel teken in een 2-dimensionale ruimte (een plat vlak) een relatie uitdrukt tussen de twee dimensies van dat vlak (oftewel X- en Y-coördinaten). Bovendien kan het teken zelf ook variëren. Deze variaties resulteren in een totaal van zes variabelen, die Bertin de grafische variabelen noemde. De grafische variabelen zijn: grootte, grijswaarde, kleur, textuur, oriëntatie en vorm. Later zijn hier door MacEachren [MacEachren, 1995] nog ordening en focus toegevoegd, bovendien verving hij de variabele kleur door kleurtint en verzadiging.

Bertin veronderstelde dat het onderscheiden en identificeren van deze variabelen op vier niveaus plaatsvindt. Hij noemde deze niveaus de waarnemingseigenschappen van de grafische variabelen. Niet alle variabelen kennen deze vier niveaus, sommige variabelen kennen slechts één niveau. De waarnemingseigenschappen zijn:

associatief Een variabele is associatief als het een directe groepering veroorzaakt tussen alle overeenkomsten die deze variabele heeft. Men noemt dit ook wel binding. Deze overeenkomsten worden over alle 'categorieën' waargenomen. Verschillende vormen van gelijke grootte en gelijke kleur zullen als groep waargenomen worden. Echter, gelijke vormen van gelijke grootte maar verschillend van kleur zullen niet als groep waargenomen worden.

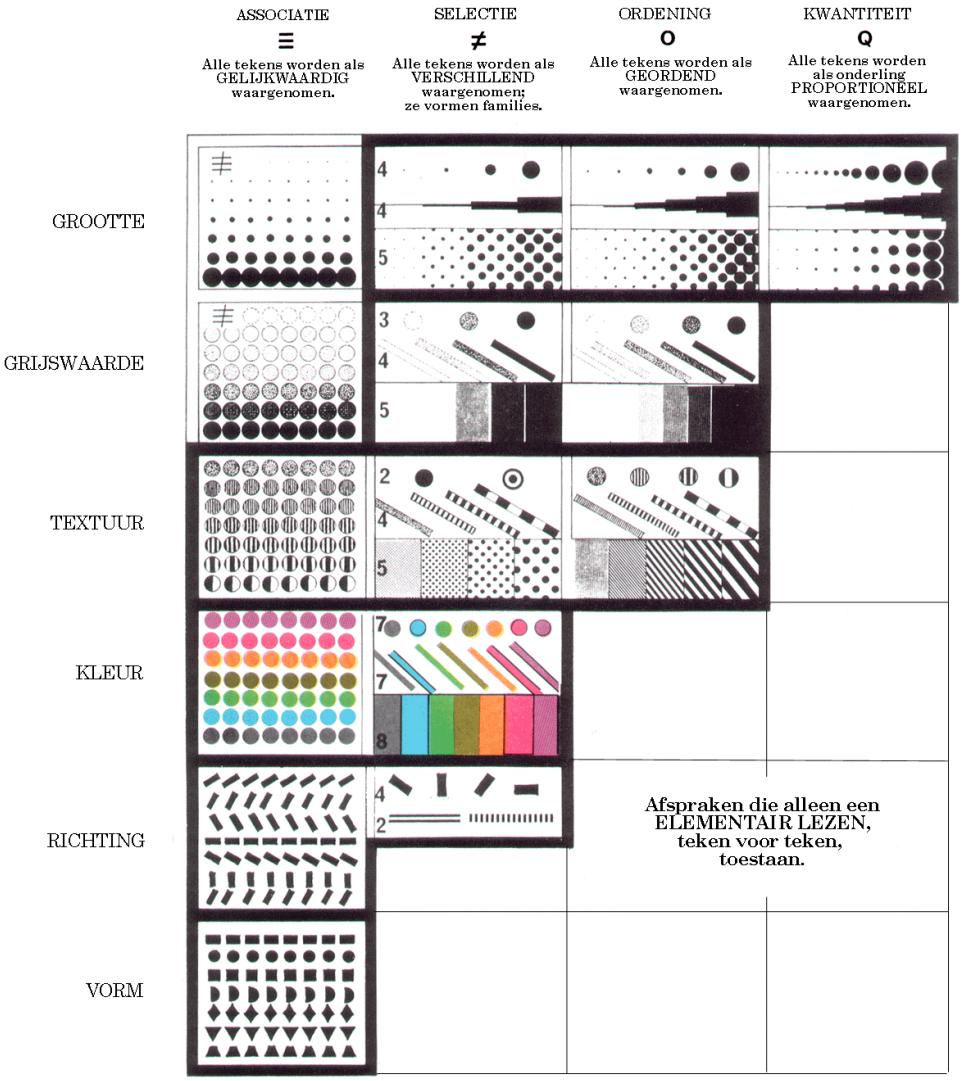
selectief Een variabele is selectief als het de waarnemer in staat stelt alle overeenkomsten tussen componenten te onderscheiden op de kaart. Men noemt dit ook wel beeldvorming. De overeenkomsten vormen een groep: een groep van rode stippen, van alle vierkanten, van alle tekens aan de rechterkant van het vlak etc.

geordend Een variabele is geordend als er direct een visuele klassificatie plaatsvindt die algemeen (niet-subjectief) is. Daarbij kan men geen absolute waardeverschillen aangeven, het één kan wel 'beter' of 'hoger' zijn dan het ander, maar hoeveel 'beter' of 'hoger' is onbekend. Voorbeelden van een geordende classificatie zijn wit-grijs-zwart, of klein-middel-groot.

kwantitatief Een variabele is kwantitatief als de visuele 'afstand' van twee elementen van een geordende component kan worden uitgedrukt in een numerieke waarde. Het gaat dus om de waardering van elementen onderling. De ene lijn is bijvoorbeeld drie keer zo lang als de andere, of deze oppervlakte is twee maal zo groot als die oppervlakte. De precisie van numerieke metingen is hierbij niet te bereiken, maar een goede schatting is altijd mogelijk.

Een overzicht van de grafische variabelen en de waarnemingseigenschappen is te vinden in figuur 2.1.

[1] De hier gebruikte literatuur uit 1983 is een Engelse vertaling van het origineel *Sémiology Grafique* uit 1967.



figuur 2.1 De samenhang tussen de visuele variabelen en de waarnemingseigenschappen [Bertin, 1983]

2.1.2 Beeldschermkartografie

Als een kaart speciaal voor het beeldscherm wordt ontworpen, moeten de kartografische regels die voor kaarten op papier gelden worden aangepast. Het ontwerpen van een beeldschermkaart kent een aantal restricties: de kartograaf heeft geen of weinig invloed op acties van de gebruiker en de configuratie van het systeem van de gebruiker. De gebruikers kunnen verscheidene beeldschermresoluties hebben, terwijl de kaart zo mogelijk geschikt moet zijn voor al deze resoluties.
Er zijn ook extra mogelijkheden: de grafische variabelen kunnen nieuwe verschijningsvormen hebben, zoals symbolen met schaduw of transparantie van objecten. Ook kunnen extra bewegende elementen als animatie worden toegevoegd, zgn. mouse-over events, pop-up menu's etc. De kaart kan dus veel meer informatie bevatten dan deze in eerste instantie weergeeft.

De schaal van een beeldschermkaart ligt meestal niet vast, omdat er ingezoomd kan worden. Er moet dus van te voren bepaald worden voor welk schaalbereik de kaart wordt ontworpen en wat er moet gebeuren als de schaal dit bereik overschrijdt.

Het gebruik van kleur op het beeldscherm verdient extra aandacht, omdat het waarnemen van kleur via een beeldscherm anders verloopt dan het waarnemen van kleur op papier. Doordat het beeldscherm het licht zelf uitzendt, kunnen kleuren dynamischer overkomen en gaan 'bewegen'. Bij een kaart op papier, waarbij de kleur wordt waargenomen door invallend licht, gebeurt dit minder snel. Bovendien kun je er niet van uit gaan dat de gebruiker van de kaart alle kleuren beschikbaar heeft in zijn systeem. Speciaal voor het internet (overigens niet speciaal voor kaarten) is het Web Safe Color Palette ontwikkeld, dat 216 kleuren bevat waarvan uit wordt gegaan dat ze bij de meeste gebruikers hetzelfde eruit zien [Rengelink, 1999].

2.1.3 Digitale Landschaps Modellen en Digitale Kartografische Modellen

Het proces dat doorlopen wordt om van de ruimtelijke objecten een visualisatie te maken, heet het kartografisch proces. Belangrijke aspecten hierbij zijn de gegevensinwinning (maar ook het bijhouden van gegevens), de bewerkingen op die gegevens en het grafisch weergeven van de gegevens. Door de nieuwste ontwikkelingen op het gebied van digitale opslag van ruimtelijke gegevens is er een tweedeling ontstaan in het kartografisch proces: Er wordt een onderscheid gemaakt tussen de gegevens zelf en de grafische weergave hiervan.
Een Geografisch Informatiesysteem (GIS) omvat beide delen: het is een systeem voor het opslaan, controleren, integreren, manipuleren, analyseren en visualiseren van geografische informatie. Het bestaat uit informatie met een geografische component, eigenschappen van deze gegevens en de benodigde, op de analyse gerichte programma's. Een belangrijke eigenschap van een GIS is dat het beslissingsondersteunend is. Voor iedere vakdiscipline die op een of andere manier met ruimtelijke gegevens werkt, kan een GIS een hulpmiddel zijn.

De fundamentele relatie tussen GIS en kartografie is dat de beschrijvingen (modellen) van de wereld worden omgezet naar beschrijvingen (modellen) van het materiele kaartbeeld. Een omgekeerde koppeling is ook mogelijk. Het verband tussen wereld en grafische voorstelling loopt altijd via onzichtbare landschapsmodellen en visualisatiemodellen. In de beeldschermkartografie spreken we van DLM (Digitaal Landschaps Model) en DKM (Digitaal Kartografisch Model) [van der Schans, 1999].

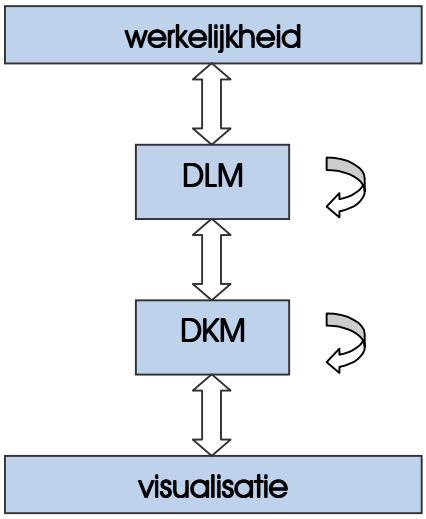
- Het **DLM** beschrijft de werkelijkheid, met alle gewenste geometrische, thematische en temporele kenmerken twee- of drie-dimensionaal, ongeacht hun grafische weergave;
- Het **DKM** beschrijft het visualisatiemedium (kaart op beeldscherm) als 2-dimensionale (dynamische) grafische beeldobjecten (stippen, strepen, vlekken, letters) met beeldgeometrie en kleuren of grijswaarden, ongeacht de semantiek.

Anders gezegd: het DLM is een model van *gedachteninhouden* en het DKM is een model van *uitdrukkingsvormen*.

2.1.4 Meta-processen en meta-informatie

Bij iedere stap in het proces van werkelijkheid naar visualisatie moeten er keuzes gemaakt worden over hoe deze stap uitgevoerd gaat worden. Deze keuzes worden meta-processen genoemd: processen die aangrijpen op andere processen. Een voorbeeld van een meta-proces is bijvoorbeeld de keuze voor het formaat waarin de gegevens worden opgeslagen. Dit heeft

invloed op de manier waarop de gegevens moeten worden ingewonnen. Beschrijvingen over de structuur van de gegevens die in het DLM voorhanden zijn wordt meta-informatie genoemd. De functie van deze 'informatie over informatie' is de relatie weergeven tussen gedachteninhouden (DLM) en uitdrukkingsvormen (DKM). Meta-informatie kan echter ook wat zeggen over de kwaliteit van de gegevens in het DLM, bijvoorbeeld met welke precisie de gegevens zijn ingewonnen. Bij iedere bewerking in het proces van werkelijkheid naar afbeeldingsvorm is de meta-informatie belangrijk om tot het volgende model te komen. Een veel voorkomende vorm van meta-informatie in kaarten zelf is de legenda. Hierin staat bijvoorbeeld dat een bepaald thema uit lijnobjecten bestaat, dat deze lijnobjecten zwart gekleurd zijn, bij welke klasse (of in welk aggregatieniveau) dit thema is ingedeeld etc.



figuur 2.2 Van werkelijkheid naar visualisatie

In figuur 2.2 is een schema te zien van de relaties tussen werkelijkheid, DLM, DKM, visualisatie, de processen die daarbij een rol spelen. De meta-informatie en meta-processen worden weergegeven door de pijlen.

2.2 GIS en internet

De vraag naar digitale kartografische producten wordt steeds groter door de grotere bekendheid en acceptatie van Informatie en Communicatie Technologie (ICT) in het algemeen en GIS technologie in het bijzonder. Genereren van digitale kartografische producten is echter arbeidsintensief, heeft hoge kosten en is moeilijk te standaardiseren. Door de toename van het gebruik van internet en de vele mogelijkheden dat het biedt, gebeurt het dat steeds meer mensen via het internet toegang hebben tot geografische informatie. Het is dus van belang dat kartografische oplossingen worden ontwikkeld met een hoge mate van automatisering en alternatieve visualisatiemogelijkheden. Kaarten van de toekomst moeten zich gemakkelijk aan kunnen passen, interactief en realistisch zijn, deze kaarten zullen het maken van betere beslissingen ondersteunen. Het belang van internet als gedistribueerd informatienetwerk is hierbij groot. Ruimtelijke data kunnen snel en eenvoudig verspreid worden. Ook kunnen digitale kaarten toegang geven tot de vele databases die via het internet te raadplegen zijn.

2.2.1 Internet GIS

Bij een internet GIS heeft een kaart meerdere functies: het geeft niet alleen inzicht in ruimtelijke relaties en patronen, maar kan ook een middel zijn om andere informatie te verkrijgen, door links naar foto's, tekst of andere kaarten op het internet. Daarnaast biedt een internet GIS de mogelijkheid kaarten dynamisch en/of interactief te maken. De interface van deze sites is vaak zo gemaakt dat de gebruiker met een simpele bewerking de gewenste informatie te zien krijgt.

Er zijn verschillende architecturen voor internet-GIS [Tuinman en van Oosterom, 1997]:
Server-side applicaties
Het GIS runt als een Common Gateway Interface (CGI)-script op een internet server. De CGI-forms geven commando's naar de server, en alleen het DKM wordt toegestuurd in de vorm van een raster in bijvoorbeeld GIF of JPEG formaat, eventueel in een door de gebruiker opgegeven symboliek. De basisgegevens en GIS-functionaliteit blijven dus bij de server en er kan geen interactie in de kaart plaatsvinden.
Client-side applicaties
Het GIS runt met behulp van een Java-applet. Het applet wordt door de browser geladen vanaf een internet-server en uitgevoerd op de computer van de gebruiker zonder installaties. Opgevraagde data wordt als DLM opgestuurd zodat de gebruiker zelf kan bepalen hoe hij de data wil visualiseren.

De meeste geografisch georiënteerde sites op internet zijn server-side applicaties. Omdat ze gemaakt zijn voor algemene doelen staan ze geen interactieve optimalisatie toe, en exploratieve grafische functies zijn ook niet mogelijk [Cecconi et. al., 1999]. De server zorgt dat alleen het DKM in de vorm van bijvoorbeeld een GIF of JPEG wordt opgestuurd zodat de grafische weergave vastligt. De interactie berust op het feit dat de server een nieuw plaatje opstuurt, maar manipulatie van de weergave is niet of nauwelijks mogelijk
Bij client-side applicaties is manipulatie wel mogelijk. Interactie leidt tot of het (lokaal) veranderen van het DLM (bijvoorbeeld bij een kleine verandering van geometrie bij generalisaties) of het veranderen van het DKM, dus een verandering van de grafische representatie.

2.2.2 Bestaande GIS-producten op internet

Tijdens dit afstudeeronderzoek is er een aantal bestaande internet-GISsen of elektronische atlassen bekeken.
De producten die bekeken en vergeleken zijn, zijn:

1. **Map Machine**: een kaartmachine van National Geographic die allerlei topografische en thematische kaarten kan genereren;
2. **HSL Atlas**: een onderdeel van de informatieverstrekking over het tracé van de toekomstige HSL, waarbij aspecten als geluid, omgeving en ruimtebeslag een rol spelen;
3. **Descartes/Kinds**: een zeer dynamisch en interactief internet GIS, waarbij de nadruk ligt op thematische kaarten;
4. **Autodesk Map Guide**: een demoversie van allerlei gemeentelijke informatie, in dit geval het monitoren van wateroverlast;
5. **Geo Data Explorer**: site van de US Geological Survey waarbij allerlei gegevens van de USGS en andere overheidsinstanties bekeken kunnen worden;
6. **ESRI Map Objects (General Map Module):** een van de vele demo's die door ESRI beschikbaar worden gesteld op het internet;
7. **Tiger Mapping**: een soort stratenboek gericht op de Verenigde Staten.

In appendix A is een screenshot van alle internet GIS producten opgenomen.

Er kan een onderscheid gemaakt worden tussen sites die een informatief karakter hebben (wat leuke kaartjes oplevert) en sites die het als doel hebben gesteld actuele en nauwkeurige informatie te leveren en belang hechten aan een goede kartografische visualisatie. De eerste

twee sites vallen onder de eerste categorie, de resterende vijf sites vallen onder de tweede categorie. Voor de onderlinge vergelijking van deze producten is gebruik gemaakt van het evaluatiemodel aangepast naar Rengelink [1999].

Vormgeving

algemene indruk Sommige kaarten maken een meer nauwkeurige indruk [6] dan anderen [1 en 2]. Het doel waarmee de kaarten worden gemaakt is hiervoor sterk bepalend, voor het gebruik door leken is het belangrijk dat de kaart herkenbaar en overzichtelijk blijft. Oriëntatie blijft voor zowel leken als professionals belangrijk.

symbolen en labels In de meeste gevallen ligt de symboolkeuze vast. Er is soms wat overlap tussen symbolen, en vooral bij de informatieve sites overlappen de tekstlabels nog wel eens [2]. Bij sommige toepassingen zijn de labels te zien als je er met de muis overheen beweegt, maar dat maakt zoeken lastig.

informatie-overdracht Er is een onderscheid te maken in *maps to see* (waarbij in één oogopslag de informatie in de kaart te zien valt) en *maps to read* (waarbij de kaart echt 'gelezen' dient te worden). Bij de laatste soort hangt het van de complexiteit van de gegevens en de kennis van de gebruiker af of de kaart op de juiste wijze wordt geinterpreteerd. Bij [3] worden bijvoorbeeld voornamelijk *maps to read* geproduceerd die voor leken onoverzichtelijk over kunnen komen. In veel van de producten (zoals [1] en [2]) worden *maps to see* gebruikt, maar de informatie-overdracht komt in het gedrang als er te veel lagen aanstaan.

klassengebruik De meeste toepassingen gebruiken voor thematische kaarten klasse-indelingen die je niet zelf in kunt stellen. Bij [3] is het juist de bedoeling zelf je klassen te bepalen en zo de optimale kaart te maken. Het is belangrijk dat er in ieder geval een default waarde bestaat voor klassenindeling.

kleurgebruik Bijna alle kaarten houden vast aan kleurconventies en zien er dus ook uit als een digitale variant van de bekende papieren kaarten. [7] biedt meerdere paletten (kleurcombinaties) aan waardoor de gebruiker, zij het beperkt, zijn eigen kleuren kan kiezen.

laagvolgorde De meeste kaarten gaan goed om met laagvolgorde, problemen ontstaan pas als er te veel lagen tegelijk aanstaan. Meestal is het niet mogelijk om zelf meer lagen aan te zetten, bij [5] kan dit echter onbeperkt.

keuze grafische variabelen Dit gaat bij alle producten meestal goed omdat er duidelijke richtlijnen zijn welke grafische variabelen (in combinatie met de waarnemings-eigenschappen) voor welk soort toepassingen geschikt zijn.

generalisatie Juist omdat deze kaarten als enige interactie pannen en zoomen heeft is generalisatie een belangrijk onderdeel. De meeste kaarten hebben een beperkt schaalbereik, waardoor er niet gegeneraliseerd hoeft te worden. Soms moet dat eigenlijk wel, en gebeurt het niet [2]. Bij [1] worden er soms lagen uitgezet bij overschrijding van een bepaalde schaal. De enige die symbolen schaalt bij in- en uitzoomen is [5].

Gebruikers Interface

interface De gebruikersinterface ziet er voor alle producten ongeveer hetzelfde uit, en aangezien de mogelijkheden beperkt zijn is de functie van de knoppen meteen duidelijk. Niet overal was een overzichtskaart aanwezig, wat bij inzoomen wel prettig is.

zoomfunctionaliteit Bij veel toepassingen kan je de zoomfactor zelf instellen, niet altijd kan er een zoomwindow getrokken worden. Bij het laatste is het meteen duidelijk hoe diep je gaat inzoomen.

schaalweergave De schaal wordt niet altijd weergegeven, terwijl dat toch een belangrijk onderdeel is in een kaart. Voor beeldschermkaarten is het belangrijk een schaalbalk te gebruiken die mee kan schalen met het toepassingswindow zelf.
legenda/aanpassing In de meeste gevallen is de legenda een opsomming van lagen met een teken erbij of ze wel of niet aanstaan. Bij de professionelere sites [3 en 6] is de legenda wat uitgebreider en kunnen de eigenschappen van de lagen (zoals kleur) via de legenda veranderd worden.

Bevraging

De meeste bevragingen zijn gericht op het zoeken van een lokatie. Bevragen, gericht op analyse, kan alleen met [3] door middel van selecties uit een gegevenstabel. Bij enkele toepassingen kunnen wel attribuuttabellen opgevraagd worden. Over het algemeen geldt dat de toepassingen niet gericht zijn op analyse en dus een beperkte queryfunctionaliteit hebben.

2.3 GIS met gedistribueerde bronnen

Het nadeel van de huidige internet GISsen is dat ze afhankelijk zijn van de aanwezige dataset bij de server. De meeste producten zijn server-side, dus de GIS functionaliteit blijft bij de server. Meestal kan er ook maar één thema tegelijk bekeken worden. Via internet moet het echter mogelijk zijn om verschillende datasets van verschillende dataservers te combineren. Zo kan de gebruiker verschillende thema's tegelijk bekijken en kan hij zijn eigen mogelijkheden van GIS functionaliteit vergroten. Een internet GIS dat meerdere databronnen gebruikt wordt wel een GIS met gedistribueerde bronnen genoemd.
In deze paragraaf wordt ingegaan op de gegevensstroom die bestaat bij het gebruik van meerdere bronnen, de catalogusdiensten die beschrijven welke gegevens er aangeboden worden, en de standaards die in ontwikkeling zijn (zoals Web Mapping Testbed en XML).

2.3.1 Gebruik van meerdere bronnen

Als een GIS op internet meerdere dataservers kan gebruiken, is de gebruiker niet meer afhankelijk van de gegevens die één enkele bron aanbiedt. Het gebruik van verschillende bronnen op het internet, of gedistribueerde databases, zorgt echter voor visualisatieproblemen. Iedere bron of database heeft zijn eigen DKM dat voor deze enkele bron het beste is. Vaak is dit model ook gebaseerd op een heersende conventie, water wordt bijvoorbeeld in de meeste gevallen met blauw aangeduid. Het is echter niet zo dat de combinatie van visualisaties van verschillende bronnen dan ook juist is. Een voorbeeld kan zijn dat het Kadaster de gebouwen bijvoorbeeld standaard met de kleur rood visualiseert, het nutsbedrijf gebruikt de kleur rood bijvoorbeeld standaard voor telefoonkabels. De aparte visualisaties kunnen dan prima zijn, in een gecombineerd beeld is het onderscheid maar moeilijk te maken. Daarbij kan het gebeuren dat door een teveel aan kaartlagen van verschillende bronnen de hoeveelheid informatie in het totale beeld zo groot wordt dat men het overzicht verliest. De visualisatie van het geheel kan dus slecht uitpakken, terwijl de afzonderlijke componenten wel goed zijn.

Het is dus wenselijk een model in het GIS in te bouwen dat:

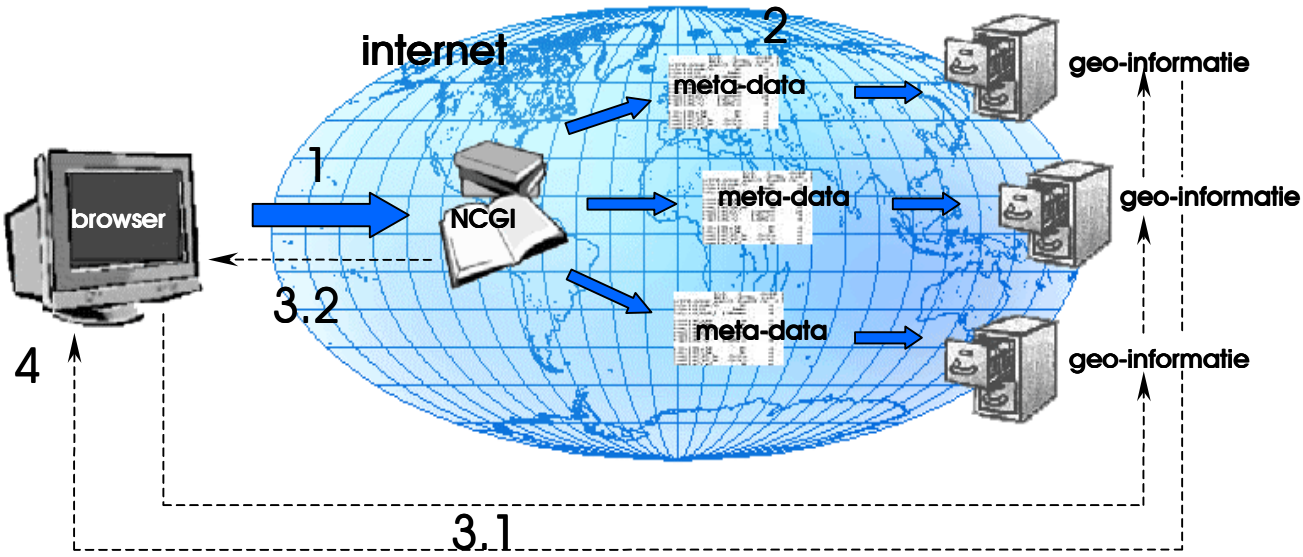
- De DKM aanwijzingen van de originele bronnen (zoals kleuren) zoveel mogelijk behoudt, dit ook in verband met de heersende conventies van weergave. Bij visualisatieconflicten moet het DKM echter alternatieven aanbieden, zonder de

eigenschappen van het DKM te veel te veranderen. Dit kan met het gebruik van verschillende voorkeursstijlen binnen het oorspronkelijke DKM;

- Om kan gaan met een teveel aan gegevens, dus kan generaliseren en waarschuwingen geeft als er een gevaar is dat de informatie-overdracht onvoldoende wordt door te veel kaartlagen;
- De eindbeslissing ook geheel aan de gebruiker kan overlaten als deze dat wenst. De nadruk moet liggen bij begeleiding en advisering van de gebruiker, maar het moet mogelijk kunnen zijn het geadviseerd DKM aan te passen tot een gebruikers-DKM.

2.3.2 Gegevensstroom van een internet GIS met meerdere bronnen

Bij het gebruiken van een multi-server internet GIS bestaat er de volgende gegevensstroom (zie figuur 2.3):



figuur 2.3 Gegevensstroom bij een multi-bron internet GIS

1. De gebruiker bepaalt het doel van de kaart en dus de vraag naar gegevens;
2. De browser zoekt naar de juiste geo-informatie server via een catalogus (in Nederland is dit de NCGI, zie § 2.3.3), die bestaat uit meta-informatie van de beschikbare geo-informatie uit verschillende bronnen;
3. Als uit de meta-informatie blijkt dat de geo-informatie die daarbij hoort voldoet aan de eisen van de gebruiker wordt de geo-informatie opgehaald. Dat kan op twee manieren: (1) de meta-informatie geeft de locatie van de geo-informatie door aan de gebruiker zodat deze de geo-informatie daar kan ophalen, of (2) via de meta-informatie wordt de geo-informatie direct opgehaald en naar de gebruiker gestuurd. In § 2.3.4 wordt de standaardisatie van gegevensuitwisseling behandeld. § 2.3.5 gaat in op nieuwe mogelijkheden van gegevensuitwisseling, die in het geval van multi-server GIS belangrijk kunnen zijn;
4. Nu moeten de gegevens gevisualiseerd worden. Het is belangrijk dat de gebruiker wordt afgeschermd voor kartografisch onverantwoorde kaarten. Doorgaans worden belangrijke beslissingen genomen aan de hand van deze visualisaties, het is dan niet alleen belangrijk dat de weergegeven informatie juist is, maar dat ook de informatie-overdracht correct verloopt. Het ontwerpen van standaard kartografische regels en het adviseren van de

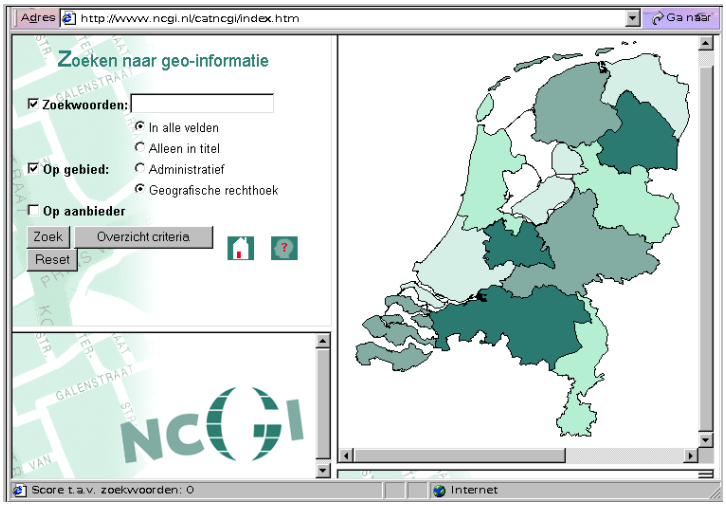
gebruiker voor het gebruik van deze gegevens is noodzakelijk om te komen tot een verantwoord eindproduct. Meer over de specifieke kartografische aspecten van een internet GIS in § 2.4.

Door selecties te maken in de gevisualiseerde gegevens in het GIS wordt stap 4, de visualisatie, opnieuw uitgevoerd en moeten de regels opnieuw worden toegepast.

2.3.3 Catalogusdiensten

Als een gebruiker van een internet GIS gegevens op wil vragen, is het handig te weten wie wat waar heeft. Speciaal voor dit doel zijn er standaards ontwikkeld voor geografische catalogi, die bestaan uit meta-informatie van de gegevens die de verschillende bronnen aanbieden. De meta-informatie beschrijft de bestanden met onder andere titel, bronhouder, oorspronkelijke schaal, bedekt gebied, maar ook actualiteit en nauwkeurigheid (kwaliteit van de gegevens). De manier waarop de meta-informatie de data beschrijft is door de CEN gestandaardiseerd. Door het zoeken in de meta-informatie kunnen de juiste gegevens en de bijbehorende bron gevonden worden.
Een geografische catalogus kan op zich zelf staan maar kan ook opgenomen worden in een centrale catalogus, een soort verzamelbestand. Hierdoor kunnen ook andere catalogi worden geraadpleegd, zoals die van andere organisaties. De communicatie tussen de centrale catalogus en de afzonderlijke catalogi is dan van cruciaal belang. Deze communicatie wordt beschreven in de nieuwe standaard van het OpenGIS Consortium (OGC, zie ook § 2.3.3): de Catalog Service. Het (Nederlandse) Nationaal Clearinghouse Geo-Informatie (NCGI) ontwikkelt een architectuur die voldoet aan deze OGC-normen. Huidige aanbieders zijn onder andere Alterra (voorheen DLO Staring Centrum (SC-DLO)), het Kadaster, diverse ministeries en provinciale overheden, DG Rijkswaterstaat en de Topografische Dienst [NCGI (internet)].

De werking van de NCGI architectuur zoals deze in een pilot is ontwikkeld is als volgt:
Een gebruiker maakt via het internet contact met het NCGI en krijgt de zoekinterface van het NCGI (zie figuur 2.4). De zoekopdracht van de gebruiker wordt vervolgens verstuurd naar het NCGI waar deze vraag opgevangen wordt door een OpenGIS Catalog Service (software). Deze catalog service van het NCGI is verbonden met de aanbieders. Zij vormen de nodes (OpenGIS Catalog) die bij het NCGI zijn aangemeld. Deze nodes kunnen op hun beurt weer doorverwijzen naar catalog services binnen de organisaties. Zo onstaat een heel netwerk van nodes van catalog services die aan elkaar geknoopt zijn. De vragen die aan de catalog service van het NCGI worden gesteld worden dus doorgesluisd en als het ware parallel afgehandeld door de verschillende nodes (of sub nodes of sub-sub nodes etc.). Uiteindelijk wordt de vraag door één of meerdere nodes (aanbieders van metadata) beantwoord. Het antwoord vindt zijn weg terug over de verschillende catalog services.



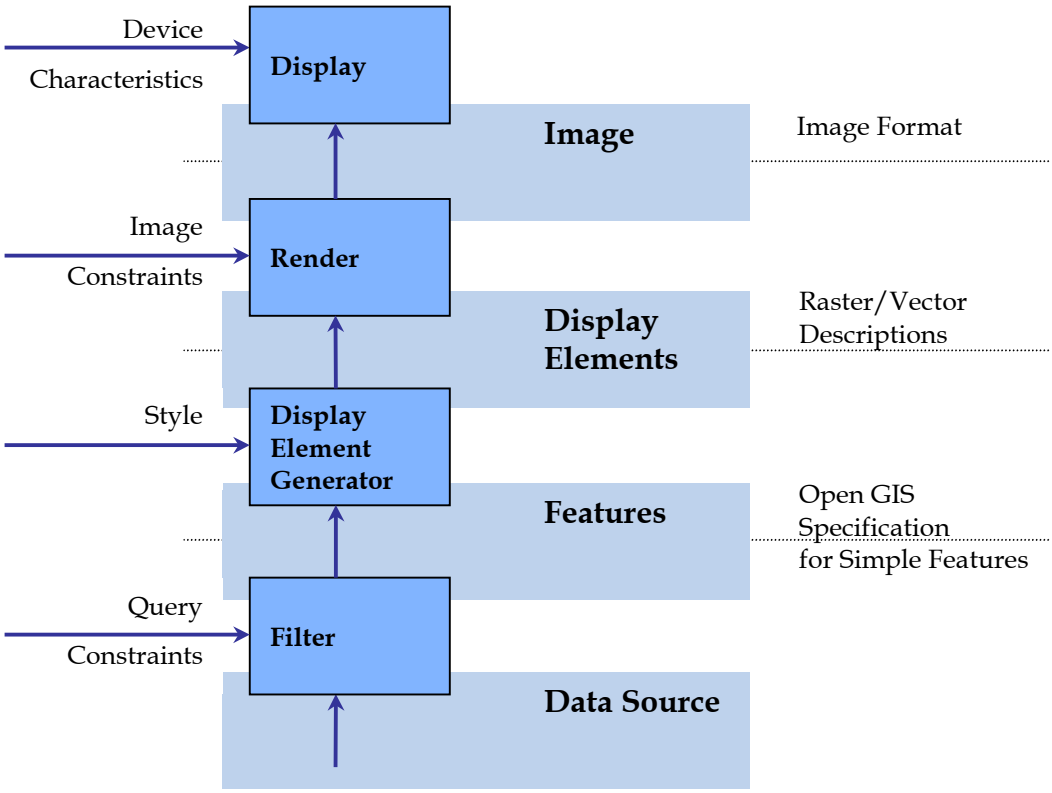
figuur 2.4: De zoekinterface van het NCGI

2.3.4 Open GIS

Open betekent dat informatie tussen verschillende GIS-sen probleemloos kan worden uitgewisseld. Voor kaarten op het web hebben veel GIS leveranciers hun eigen mapping tool (zoals Autodesk MapGuide en Intergraph GeoMedia). De onderlinge communicatie en gegevensuitwisseling is helaas niet optimaal door deze eigen ontwikkelingen. Om de samenwerking, of interoperabiliteit te bevorderen heeft het OpenGISConsortium (OGC) het WebMappingTestbed (WMT) opgezet waarbij de systemen wel kunnen communiceren. [Doyle, 2000]

Een Web Map Server kan drie dingen, gestandaardiseerd door het OGC in drie type vragen:
1. GetMap (een kaart maken of samenstellen);
2. GetFeatureInfo (basisvragen over de inhoud beantwoorden);
3. GetCapabilities (andere programma's vertellen wat hij kan doen).

Web mapping gaat ervan uit dat de standaard web browser vragen kan stellen aan de Map Server in de vorm van Uniform Resource Locators (URLs). De informatie kan aan verschillende Map Servers gevraagd worden. Door het opgeven van een ruimtelijke context (x- en y-waarden waarbinnen de kaart opgevraagd wordt) is het mogelijk twee verschillende kaarten te combineren. Als een van deze kaarten bevraagbaar is, kan de gebruiker meer informatie opvragen door bijvoorbeeld een pixel (x en y offset) op te geven in de URL. Omdat iedere Map Server verschillende soorten informatie heeft waar verschillende kaarten uit gegenereerd kunnen worden, moet het mogelijk zijn een lijst van de mogelijkheden van de betreffende Map Server beschikbaar te stellen. Dit schept mogelijkheden voor een bevraagbare catalogus die terug kunnen verwijzen naar de Map Servers.



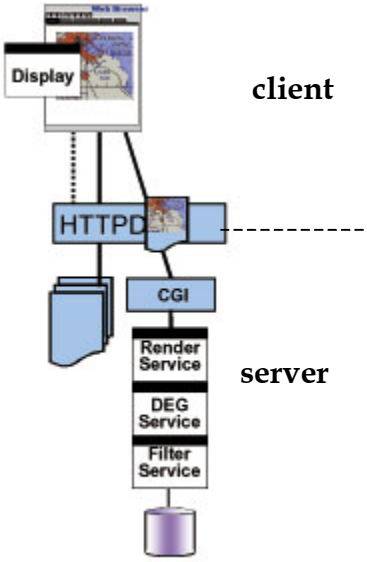
figuur 2.5: gestandaardiseerd weergaveproces (naar OpenGIS WebMap Specifications [Doyle, 2000])

Het weergaveproces (zie figuur 2.5) kent vier afzonderlijke fases (data-source, features, display elements, image) die zich voordoen als gegevens getransformeerd worden van de gegevensbron naar het afbeeldingsmedium waarop de gebruiker de kaart te zien krijgt. De bewerkingen die leiden tot een nieuwe fase kunnen, afhankelijk van de architectuur, zowel bij de server als de client plaatsvinden. Als de meeste bewerkingen zich aan de serverkant bevinden, is het resultaat van deze bewerkingen aan meer beperkingen (constraints) gebonden dan als de client deze bewerkingen zelf uitvoert. Als de server bijvoorbeeld opeenvolgend filtert, weergave-elementen genereert en rendert is het resultaat beperkt tot een afbeelding in bijvoorbeeld GIF formaat. Zou de server alleen filteren, heeft de client de vrijheid deze naar eigen voorkeur af te beelden.

Web Mapping kan uitgelegd worden in termen van informatie-overdracht: Welke informatie gaat van webserver naar de gebruikerscomputer en hoe is die informatie verpakt? Dit heeft geleid tot drie vormen (architecturen): de picture case, de graphic case en de data-case.

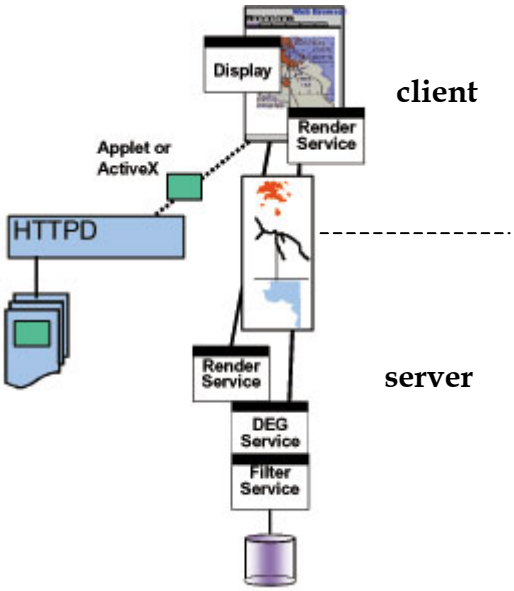
1 Picture case

Op aanvraag van de gebruiker wordt er door de webserver een plaatje van de kaart, meestal in de vorm van een GIF of JPEG, gemaakt. Het plaatje is gemaakt door de MapServer;



figuur 2.6.1 Picture case

[Doyle, 2000]



figuur 2.6.2 Graphic element case

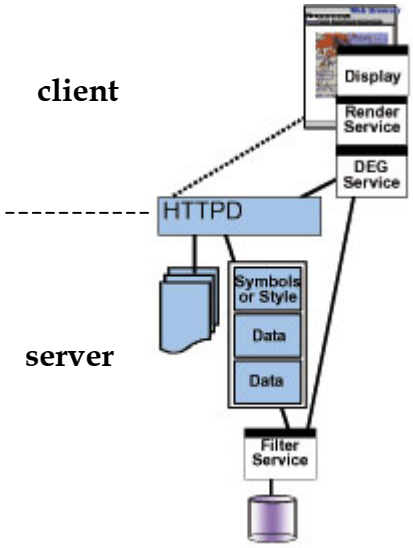
[Doyle, 2000]

2 Graphic Element Case

De Webserver stuurt een verpakte set van individuele grafische elementen in een bepaalde projectie en referentiesysteem. De grafische weergave ligt dus vast, een weg wordt dan voorgesteld door een polylijn van 2 pixels dik, kleur rood, een meer wordt voorgesteld door een blauwe polygoon, etc. Sommige grafische elementen zijn zelf een voorgemaakt plaatje, dus de picture case kan een subset zijn van de Graphic Element Case;

3 Data case

In deze vorm wordt geografische feature data van de server naar de gebruiker verzonden. Hierbij wordt gebruik gemaakt van Open GIS Simple Features in eXtended Markup Language (XML, zie § 2.3.5) voor de gegevens zelf en eXtended Stylesheet Language Transformation (XSLT, zie § 2.3.5) om visualisatievoorschriften mee te zenden met de gegevens. Met behulp van deze 'labels' is de gebruiker in staat de verschillende typen gegevens te onderscheiden om een goede kaart te maken.



figuur 2.6.3 Data case [Doyle, 2000]

Voor het gebruiken van meerdere data-bronnen waarbij het DLM wordt opgestuurd is alleen de laatste architectuur (data-case) van belang, omdat dit de client de mogelijkheid biedt de data uit verscheidene bronnen te combineren en te visualiseren aan de hand van de voorkeuren die de bronnen met de gegevens aanleveren.

In het vervolg van dit afstudeeronderzoek gaan we er dus ook vanuit dat de gegevens in de 'data-case' worden verzonden.

2.3.5 XML, GML en standaardisatie

Een van de meest recente ontwikkelingen op het gebied van communicatie-standaardisering is eXtensible Markup Language (XML) [Bray et.al., 2000]. XML is een nieuwe opzet voor gegevens op Internet dat vooral van belang kan zijn voor het beheren van grote aantallen complexe gegevens op het Internet. HTML is beperkt als het gaat om semantiek, de betekenis van de gegevens en de verstuurde informatie. XML kan de gegevens op een gestructureerde en betekenisvolle manier weergeven. Door de semantisch rijke *markup* is een XML file makkelijk te lezen en spreekt de code voor zich. Door de wereldwijde acceptatie van XML op het internet zijn alle benodigde tools vrij verkrijgbaar op internet.
XML gaat over zowel data als meta-data (informatie over informatie) en het idee dat bepaalde groepen mensen dezelfde behoeften hebben om de data die zij gebruiken te beschrijven en te organiseren. XML is, net als HTML, een verzameling van labels en declaraties. Het beperkt zich echter niet tot het plaatsen van gegevens op een webpagina, maar het richt zich ook op het verstrekken van informatie over deze gegevens en welke relatie zij hebben tot andere gegevens. Waar HTML wordt gebruikt als het weergaveformaat, kan XML dienen als het opslag- en uitwisselingsformaat van het complete gegevensmodel.

XML kent twee vormen: well-formed en valid. *Well-formed* is de eenvoudige standaard en vereist dat een document een XML-proloog heeft, dat alle attributen netjes genest zijn en dat alle labels een begin- en eindmarkering hebben. Alle XML documenten moeten in ieder geval *well-formed* zijn. Als een HTML document netjes is opgesteld lijkt dat al behoorlijk op deze XML-vorm.
Een voorbeeld van een well-formed XML fragment:

```
<IMG SRC="mijnplaatje.gif" BORDER=1 HEIGHT=55 WIDTH=40 ALT="Mijn Plaatje"></IMG>
```
↑begintag eindtag ↑

Valid documenten zijn uitgebreider dan well-formed documenten omdat ze vergezeld worden van Document Type Declarations (DTD) dat de structuur van het document formeel vastlegt. Het DTD kan een deel van het document zelf zijn of in een apart document opgeslagen zijn dat via een verwijzing te vinden is. Over het algemeen worden erg complexe DTD's in een apart document bewaard. Een DTD is eenvoudig gezegd een beschrijving van elementen, entiteiten en attribuut-declaraties in een gesimplificeerde SGML (Structured Geography Markup Language) declaratiestijl.

XML documenten hoeven niet vergezeld te worden door een DTD. Een well-formed document kan de markup elementen definiëren op het moment dat ze gebruikt worden. Als het XML document gelezen wordt, moet wel aangekondigd worden dat er geen DTD is. Het voordeel hiervan is dat er geen DTD gemaakt hoeft te worden (wat erg complex kan zijn), het nadeel is echter dat er geen automatische controle is op een goede structuur van het document. Voor complexe documenten heeft het de voorkeur wel een DTD te definiëren.
Deze kleine aanpassingen in de wereld van de mark-up talen zorgen ervoor dat verspreiders van data niet meer afhankelijk zijn van HTML codes die niet altijd nauwkeurig zijn en vele eindlabels missen. Voordat een document zichzelf well-formed kan noemen moet het aan een aantal minimum eisen voldoen. Het verwacht dus meer van de makers van de documenten, maar maakt het voor programmeurs makkelijk om meer betrouwbare systemen te maken met minder moeite dan voorheen.

De toepassingen die de acceptatie van XML zullen versnellen, zijn juist die toepassingen die niet bereikt kunnen worden met de beperkingen van HTML. Deze toepassingen op het Internet kunnen worden onderverdeeld in vier groepen:
1. Toepassingen waarbij de gebruiker twee of meer heterogene databases gebruikt;
2. Toepassingen die een behoorlijk deel van de verwerking van de gegevens tussen server en gebruiker verspreiden;
3. Toepassingen die dezelfde data voor verschillende gebruikers op een andere manier afbeeldt;
4. Toepassingen die op maat gesneden informatie aanbiedt aan de hand van de eisen van de verschillende gebruikers.

In het geval van een internet GIS met gedistribueerde bronnen lijkt de toepassing van XML dus een middel bij uitstek, aangezien alle vier de groepen van toepassing zijn.
Voor alle hierboven genoemde toepassingen is er op het moment een alternatief beschikbaar in de vorm van scripts die in het HTML document geplaatst kunnen worden. Deze kunnen met behulp van plug-ins en Java applets gebruikt worden. De filosofie achter XML is dat de gegevens bij de bron horen en dat gebruikers het beste af zijn met een gestandaardiseerde toepassing, dus niet afhankelijk zijn van bepaalde script-talen en aanbieders op het internet.
Speciaal voor geografische objecten is er Geography Markup Language (GML), een XML codering die specifiek gericht is op de uitwisseling van geografische informatie. Deze codering transporteert en bewaart de geografische informatie, zowel de geometrie als de eigenschappen van geografische objecten. Het is te verwachten dat de uitwisseling van geografische informatie in de toekomst gestandaardiseerd wordt met GML als uitgangspunt.

XSL en XSLT

Zoals eerder gezegd is er, naast de levering van de gegevens zelf, behoefte aan een beschrijving van de voorkeursweergave van de gegevens. Ook hier biedt XML een veelbelovende mogelijkheid in de vorm van stylesheets. Na het ontwikkelen van een Extensible Stylesheet Language (XSL) voor XML, is er nu een specificatie voor het

transformeren van XML documenten, XSLTransformations (XSLT). Dit is ontwikkeld om XML-documenten te transformeren naar een specifieke presentatie-syntaxis. Het kan dus zeer geschikt zijn om naast de data in GML ook een XSLT met presentatievoorschriften mee te sturen. Het XSLT-gerelateerde onderzoek bij het Finse Geodetische Instituut [Letho, 2000] concentreert zich op de generalisatie van XML-gecodeerde ruimtelijke gegevens. De eerste resultaten van het onderzoek laten zien dat XSLT gebruikt kan worden als een generalisatie werktuig door de gegevens op een goede manier te filteren, of door te selecteren uit andere representaties van de geometrie, voor zover aanwezig in de dataset. Zo is het mogelijk om bepaalde thema's in de dataset te benadrukken, bijvoorbeeld alle overheidsgebouwen.

Het volgende voorbeeld van een gedeelte van een XSLT file laat zien dat een object met de code 'overheid' (governmental) bij de transformatie anders wordt behandeld dan gebouwen die niet die code hebben [Letho, 2000]:

```
<xsl : template match = "Feature [@featureType = 'building']">
    <xsl : if test = "Description [text () ="governmental"]>
        <g style = "fill : red; stroke : black; stroke-width : 2">
            <xsl : apply-templates select = "Polygon"/>
        </g>
    </xsl : if>
    <xsl : if test = "Description [not text () ="governmental"]>
        <g style = "stroke : red ; stroke-width : 1">
            <xsl : apply-templates select = "Point"/>
        </g>
    </xsl : if>
</xsl : template>
```

Op het moment zijn er veel specificaties in ontwikkeling voor het uitwisselen van gegevens op het internet. Vooral bij het ontwerpen van geografische toepassingen zijn deze specificaties erg belangrijk. XML wordt gezien als een toekomstige technologie die nieuwe mogelijkheden voor het werken op het internet introduceert. XML zal niet alleen op de levering van data veel invloed uitoefenen, maar ook op het uitwisselen van willekeurige datasets. Er zijn ondertussen al vele XML-vocabulaires zoals XML Schema en XLink ontwikkeld voor verschillende toepassingsdomeinen [Letho, 2000].
Verwacht wordt dat XML de nieuwe standaard wordt op het gebied van het gecodeerd verzenden van data over het internet. Juist geografische toepassingen zullen baat hebben bij de XML-ontwikkelingen, vooral bij het vervangen van het lang gewaardeerde DTD dat echter niet geschikt is voor het coderen van de data zelf. Het Open GIS Consortium ontwikkelt een vocabulaire voor geografische gegevens binnen het Web Mapping Testbed. In dit afstudeeronderzoek wordt nog geen gebruik gemaakt van GML en XSLT, maar de verwachting is dat dit in vervolgonderzoek zeker een grote rol gaat spelen.

2.4 Kartografische aspecten van een internet GIS

Het gebruik van meerdere databronnen heeft dus gevolgen voor de kartografische regelgeving en het opstellen van een DKM. Als een kartograaf een kaart ontwerpt, moet hij goed op de hoogte zijn van de aard van de gegevens, het gebruik van variabelen, het effect van combinaties en het verkrijgen van een juiste visuele hiërarchie. Het doel van de kaart is daarbij het belangrijkste gegeven. Het gebruik van een GIS op internet impliceert dat de kartografische kennis niet altijd aanwezig is. Om toch een goede kaart op het beeldscherm te krijgen en tegelijkertijd de gegevens van deze kaart te kunnen analyseren moeten de algemene kaartontwerpregels worden samengevoegd in de vorm van een Kartografisch

Expertsysteem. Deze set van regels vormt van de verschillende kartografische modellen van de bronnen een geïntegreerd DKM, zodat het KES de gebruiker kan adviseren en begeleiden tot een kartografisch verantwoord product. De gebruiker is hierbij sturend door zelf aan te geven welke gegevens het belangrijkst zijn om het doel van zijn visualisatie te bereiken.

Een Internet GIS is onder te verdelen in drie delen [aangepast naar Rengelink, 1999]:

1. het grafische model;

Dit is het KES: onder dit deel vallen alle denkbare kartografische regels en heeft betrekking op het uiterlijk van de kaarten en de informatie-overdracht die kaarten kunnen bieden.

- Labels en symbolen moeten de juiste grootte hebben en elkaar niet overlappen;
- Niet te weinig en niet te veel informatie in het kaartbeeld;
- De gekozen kleuren moeten overeenkomen met het onderwerp (conventies) en niet door een andere kaartlaag in gebruik zijn.

2. de gebruikers-interface;

Onderdelen die het KES in werking stellen: hieronder vallen alle regels die betrekking hebben op elementen rondom de kaart, onder andere de interface en de interactie-mogelijkheden.

- De mogelijkheden moeten direct duidelijk zijn;
- Juist gebruik van zoom-functionaliteit;
- Een overzichtskaart moet aanwezig zijn of opgehaald kunnen worden, deze kaart moet consistent blijven met de actieve kaart.

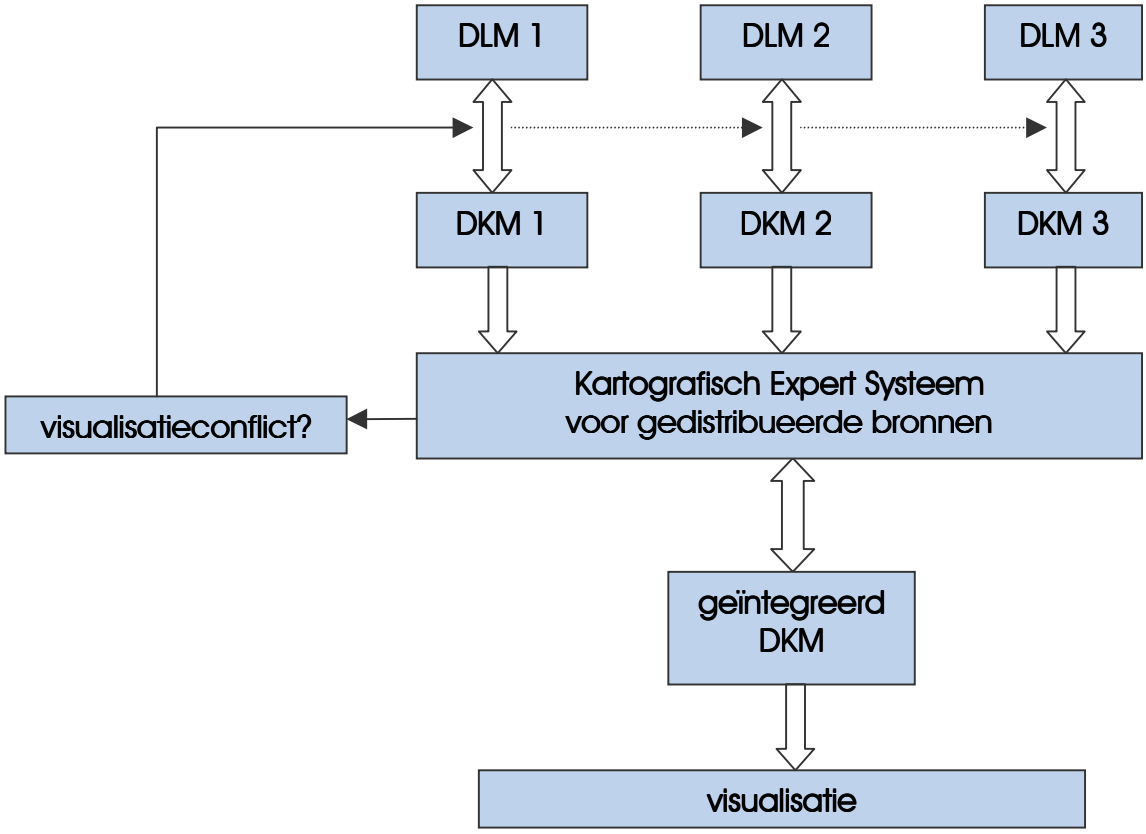
3. De query-functionaliteiten;

Dit deel van de regels staat apart, omdat het bevragen en selecteren van de informatie op basis van attributen een belangrijk onderdeel en kenmerk van een GIS is. Selecties van de gebruiker geven aan dat hij deze selecties belangrijk vindt, het KES moet zijn regels dan opnieuw toepassen.

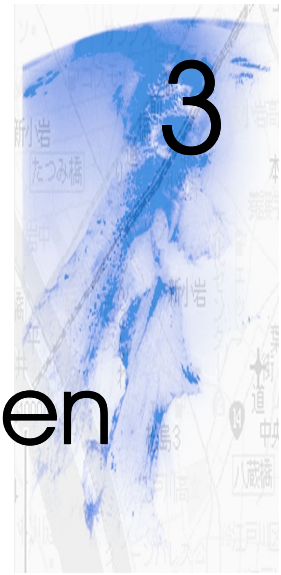
- Het type vragen dat gesteld kan worden moet duidelijk zijn (wat, waar, hoe, wat als, welke relatie, wat is veranderd);
- Maak gebruik van het bereik van de aanwezige attributen bij het begeleiden van vragen, zodat foute vragen kunnen worden ondervangen.

Vooral de regels van het grafische model kunnen goed geïmplementeerd worden in een DKM. Het is van belang te bedenken dat het model toegepast moet worden op het internet en dat voor beeldschermkaarten andere regels gelden (zie § 2.3.2).

In figuur 2.8 is te zien wat de samenhang is tussen de verschillende DLMs en DKMs van een multi-bron GIS en de plaats van het KES in het geheel.
Iedere bron die de data aanlevert doet in de vorm van een DLM (DLM1, DLM2, DLM3), inclusief de voorkeuren van de bron voor het DKM (DKM1, DKM2, DKM3). Deze voorkeur is ontstaan doordat de bron zijn eigen regels heeft toegepast op het DLM. De gegevens worden samengevoegd waarbij het KES de regels gaat toepassen. Op het moment dat er een conflictsituatie ontstaat in de presentatie, wordt bekeken welke gegevens een aangepaste weergave krijgen en welke (tweede) voorkeur de gegevensbron geeft. Als er daarna geen conflicten meer worden geconstateerd, vormt het geïntegreerde DKM voor een juiste visualisatie.



figuur 2.7 De samenhang tussen de verschillende DLMs en DKMs en de rol van het Kartografisch Expertsysteem.



3 Expertsystemen

Door het grootschalige gebruik van computers voor allerlei doeleinden is er in de loop van de tijd behoefte gekomen aan een grotere mate van automatisering van allerlei taken. Voor sommige taken is het niet moeilijk om kennis in de computer in te voeren zodat het systeem 'intelligente' taken kan uitvoeren. Simpele wiskundige bewerkingen bijvoorbeeld zijn gemakkelijk in programmaregels te vatten en zullen geen problemen opleveren.
Het wordt anders als de regels niet zo duidelijk voortkomen uit de bewerkingen. Routinematige handelingen die experts in de praktijk doen zijn vaak gebaseerd op een menselijke redenering, zoals heuristische regels, praktijkervaringen en verwachtingen. De computer redeneert op basis van deze menselijke redenering, maar alleen in het geval dat deze menselijke redenering in regels of kennis te vatten is. Verder kunnen er ook andere dingen meespelen, zoals het intuïtieve handelen of associatief denken. Het is zelfs voor psychologen nog niet mogelijk deze redeneringen volledig te bevatten, zodat het niet reëel is om deze redeneringen als regels in de huidige kennissystemen te implementeren.
In § 3.1 wordt een algemene definitie gegeven van een expertsysteem, waarna er specifiek wordt ingegaan op kartografische expertsystemen voor een internet GIS. Het kaartontwerp en een stappenplan voor kaartontwerp worden in § 3.2 behandeld. In dezelfde paragraaf wordt een aantal aandachtspunten opgesteld die van belang zijn bij het opstellen van regels voor een KES. Op deze aandachtspunten wordt nader ingegaan in § 3.3. In § 3.4 worden de regels zelf opgesteld aan de hand van een stappenplan van het kaartontwerpproces.

3.1 Definitie van een expertsysteem

De definities van expert systemen of kennissystemen variëren naargelang de functie, de structuur of beide. De meeste definities zijn gebaseerd op het vooronderstellen van regel-gebaseerde expertsystemen. Een algemene definitie van een regel-gebaseerd (of rule-based) expertsysteem kan zijn:

> Een systeem dat gebruik maakt van menselijke expertise om een set van regels te beschrijven die een bepaald probleemgebied behandelen.

Het doel van een kartografisch expertsysteem is de mogelijkheid te bieden voor het automatisch produceren van een complete kaart gebaseerd op de aanwezige gegevens uit heterogene bronnen.

Het gebruik van rule-based expertsystemen heeft zowel voordelen als nadelen:

Voordelen
- Het systeem vergeet niets;
- Het systeem kan gemakkelijk gereproduceerd worden, in tegenstelling tot het 'reproduceren' van menselijke experts;
- Het systeem is efficiënt;
- Eenmaal opgezet kost het systeem veel minder dan een expert;
- Ontwikkelingskosten kunnen over meerdere gebruikers verdeeld worden;
- Het systeem is consistent en niet beïnvloedbaar;
- Het systeem documenteert al zijn beslissingen en is daarbij compleet;
- Door het samenvoegen van kennis van verschillende experts kan het systeem een brede kennis opbouwen;

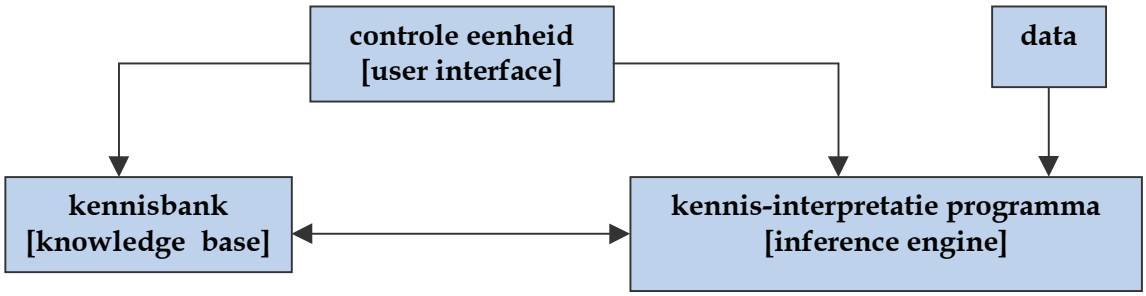
Nadelen
- Intuïtieve beslissingen die mensen maken kunnen niet zo makkelijk als regel ingevoerd worden;
- Het systeem kan niet creatief zijn;
- Het systeem kan niet of moeilijk leren;
- Het systeem kan niet of moeilijk bouwen op ervaring;
- Het systeem kan moeilijk onderkennen of er geen oplossing is voor een probleem of wanneer een probleem buiten hun expertisegebied valt.

Overigens zijn twee andere soorten expertsystemen, neurale netwerken en artificial intelligence technieken, niet gebaseerd op regels maar op patroonherkenning. Deze systemen kunnen wel leren en ervaring opbouwen.

Rule-based systemen zijn vooral gericht op de klassificatie van de gegevens en de klasses verbinden met de voorwaarden die daarop van toepassing zijn. Deze eigenschappen, in combinatie met de voordelen die het rule-based kennissysteem biedt, zorgen ervoor dat een rule-based kartografisch expertsysteem een hulpmiddel kan zijn voor het automatiseren van bepaalde taken die voorkomen bij het maken van een kaart.

3.1.1 Structuur van een rule-based expertsysteem

De structuur van een rule-based expertsysteem valt (naast de gegevens zelf) uiteen in drie delen (zie figuur 3.1):



figuur 3.1 Structuur van een rule-based expertsysteem

1 Controle-eenheid of user-interface
- is gericht op de gebruikers;
- controle bestaat uit gebruikersvoorkeuren en experts die nieuwe kennis in het systeem invoeren.

2 Kennis-interpretatieprogramma of inference engine
- legt het verband tussen feiten en acties en voert de regels uit;
- geeft de volgorde aan waarin de regels worden doorlopen.

3 Kennisbank of knowledge base
- is een verzameling van declaratieve kennis (feiten) en procedurele kennis.

Deze regels hebben dan de volgende structuur:

IF *[toepasbare voorwaarde]* THEN *[aanbeveling]*

[toepasbare voorwaarde] is dan bijvoorbeeld:
- gegevenskarakteristieken en relaties tussen gegevens;
- karakteristieken van een huidige presentatie (als er bijvoorbeeld nieuwe data wordt toegevoegd);

De voorwaarden zijn de *feiten* die in het systeem aanwezig zijn.
[aanbeveling] is dan bijvoorbeeld:
- een generieke taak die moet worden uitgevoerd;
- een verwijzing naar de volgende regel die moet worden toegepast;
- een nieuwe regel die toegevoegd wordt aan de kennisbank.

De aanbevelingen zijn de *procedurele kennis* die in het systeem aanwezig is.

3.1.2 Eigenschappen van een expertsysteem

Een ideaal expertsysteem heeft de volgende eigenschappen:
- het moet conclusies kunnen trekken op basis van onzekere gegevens;
- het moet de mogelijkheid hebben zijn eigen kennis te begrijpen en toe te passen, het moet ook conflicten kunnen herkennen;
- het moet gemaakte keuzes kunnen verklaren (ofwel er moet een logische structuur in de besluitvorming aanwezig zijn);
- het moet kunnen leren.

De eisen aan deze eigenschappen zijn nogal hoog, het zal duidelijk zijn dat de ontwikkeling van een ideaal expertsysteem iets is wat nogal wat tijd en inspanning vereist. Vooral (menselijke) intelligentie in het systeem en de mogelijkheid om te leren zijn erg moeilijk te implementeren. Het is belangrijk te onderkennen dat de eerste stappen van een expertsysteem alleen voor relatief simpele toepassingen gebruikt kan worden.

3.1.3 Internet GIS en kartografische expert systemen

Voor een internet-GIS dat met meerdere bronnen werkt, is het werken met een kartografisch expertsysteem een goede oplossing om met de gegevens uit die bronnen, inclusief de presentatievoorschriften, een goede kaart te maken. Het gebruik van een expertsysteem heeft in dit geval drie redenen:

1. Het is onduidelijk wie de gebruikers van het internet-GIS zijn, dus het is ook niet duidelijk of de gebruiker genoeg expertise heeft om een kaart samen te stellen die aan de kartografische regels voldoet. Gebruikers met geen of weinig ervaring in het ontwerpen van kaarten moeten begeleid worden, gebruikers met meer ervaring moeten de mogelijkheid hebben deze adviezen naast zich neer te leggen;
2. In het geval van een picture case of graphical element case [zie ook §2.3.4] maakt de server de belangrijkste keuzes. Meestal weet deze server van tevoren welke data hij tot zijn beschikking heeft en kan hij dit goed afbeelden. Voor een multi-bron aanpak zijn deze mogelijkheden echter niet geschikt. Een data-case opzet is dus noodzakelijk. Het is van tevoren echter niet duidelijk welke data gebruikt gaat worden en dus zijn er geen vaste regels voor weergave. De regels uit een KES kunnen flexibel worden toegepast op het moment dat de gegevens uit de verschillende bronnen samenkomt.
3. Het gebruik van een GIS op internet vraagt om automatisering.

Het idee van een intelligent en gebruikersvriendelijk GIS impliceert dat het niet alleen een verzameling van een aantal krachtige instrumenten is die gebruikt kunnen worden voor het weergeven en analyseren van gegevens, maar dat het de gebruiker ook kan adviseren in het weergeven van de gegevens. In dit specifieke geval moet het expertsysteem dus de volgende eigenschappen hebben [naar Andrienko & Andrienko, 1999]:

1. De mogelijkheid om het doel van de gebruiker te begrijpen;
2. De mogelijkheid om de data zo weer te geven dat deze doelen bereikt kunnen worden;
3. De mogelijkheid om de gebruiker te ondersteunen in het exploreren van deze data.

Deze laatste stap impliceert dat de data niet alleen bekeken kan worden, maar dat de gebruiker de weergave kan veranderen om relevante patronen te ontdekken (interactieve visualisatie). Vooral op het gebied van thematische kaarten is dit een interessante ontwikkeling, omdat de gebruiker dan niet afhankelijk is van de keuze van de kartograaf. In topografische kaarten kan bijvoorbeeld een attribuuttabel opgevraagd worden, of alleen de percelen die groter zijn dan een bepaalde oppervlakte. De kennis die gebruikt wordt om de gebruiker te begeleiden in het samenstellen van een kaart kan als regels in het systeem geïmplementeerd worden.

In de ontwikkeling van een kartografisch expertsysteem zijn er twee belangrijke onderdelen.

Het *eerste onderdeel* is om bestaande kartografische kennis om te zetten in regel-gebaseerde kennis, dus hoe de kennis ingevoerd wordt. Bij van een kartografisch expertsysteem is het grootste probleem dat veel kennis is gebaseerd op visuele vaardigheden en associatief denken. De kartografische regels staan in zoverre niet vast dat er voor een bepaald gegeven onderwerp geen pasklaar antwoord is: er kunnen meerdere juiste varianten zijn. De kennis zal dus beïnvloed worden door het wel of niet om kunnen zetten in regels. Een optie kan zijn om meerdere keuzeregels in het kennissysteem op te nemen, waarbij de context bepaalt welke regel het meest van toepassing is.

Het *tweede onderdeel* is een leek begeleiden in het maken van een kaart, dus het gebruiken van de regels die in door het eerste onderdeel zijn opgesteld. In veel gevallen is het meest effectieve expertsysteem geconcentreerd op een advies over wat voor soort kaart van de aanwezige gegevens gemaakt kan worden, waarbij de gebruiker dan het minste aantal beslissingen hoeft te nemen. Als het algemene idee van de kaart in termen van schaal, gebied

en inhoud beschreven is, moet het systeem voorwaarden opleggen aan de onervaren gebruiker.

3.2 Het kartografisch ontwerp

De eerste stap in het ontwerpen van een kartografisch expertsysteem is een goed overzicht te krijgen van het proces dat een kartograaf doorloopt om van gegevens naar een kaart te komen. Er zijn een aantal aspecten waar bij het ontwerpen op gelet moet worden om de kaart visueel verantwoord te maken. Deze aspecten worden in § 3.2.1 beschreven. In § 3.2.2 wordt het kaartontwerp voor een internet GIS toegelicht, wat uiteindelijk in § 3.2.3 resulteert in een stappenplan voor kaartontwerp.

3.2.1 Aspecten in kaartontwerp

Bij het ontwerpen van een kaart moet de kartograaf erop letten dat de kaart visueel verantwoord is. Met dat laatste wordt bedoeld of de kaart prettig is om naar te kijken, of de kleurkeuze goed is, of het kaarttype past bij het uit te beelden fenomeen etc. Deze keuzes hangen natuurlijk af van het type kaart, zoals een topografische kaart, een leidingen kaart of een thematische kaart. Ieder type kaart kent zo zijn eigen regels en voorwaarden.
Voor het maken van de juiste keuze zijn er richtlijnen die per (type) kaart kunnen verschillen, maar wel als algemeen geldend aangemerkt kunnen worden [Jones, 1997].

helderheid en betrouwbaarheid
De kwaliteit van kaartsymbolen en tekst moet goed zijn, dat wil zeggen dat ze altijd scherp zijn in iedere beeldschermresolutie. De grootte van symbolen en grafische objecten staat niet vast maar hangt af van de pixelgrootte. Ook de onderscheidbaarheid van symbolen is van belang: bij ieder zoomniveau moeten de symbolen goed van elkaar te onderscheiden zijn. Er moeten dus regels komen over tolerantiegrenzen en het verplaatsen van symbolen (generalisatie);
hiërarchische structuur
Soms worden verschillende thema's weergegeven die hun eigen hiërarchische onderverdeling hebben.
Er zijn drie typen hiërarchische verdelingen:

1. *spaghetti-structuur*: alleen onderscheid in lijn- en puntsymbolen, bijvoorbeeld de lijndikte of waarde;
2. *partitie structuur*: onderverdeling van bijvoorbeeld oppervlakte-objecten:
   - verschillende categorieën worden onderscheidbaar door bijvoorbeeld het gebruik van verschillende kleuren;
   - binnen categorieën subcategorieën, door bijvoorbeeld de variatie van lichtheid van die kleur;
3. *3D structuur*: veranderen van de visuele waarde van de component, bijvoorbeeld de ondergrond in grijs afbeelden, belangrijke dingen in kleur, of de kleurverzadiging aanpassen naargelang de belangrijkheid;

kleur, patroon, visueel contrast en balans van het visuele geheel
Dit beïnvloedt de helderheid en betrouwbaarheid, doordat het gebruik van niet contrasterende kleuren kan leiden tot een slechte onderscheidbaarheid. Of dit het geval is hangt dan ook weer af van de grootte van het symbool of object. Het is echter belangrijk niet

te veel verschillende kleuren te gebruiken met het oog op de balans van het visuele geheel. De visuele variabelen van Bertin [zie §2.1.1] en de regels die hierbij horen zijn zeker gedeeltelijk ook bruikbaar voor beeldschermkartografie;

figuren en ondergrond

De verscheidenheid van het gebruik van grafische variabelen op het beeldscherm kan ervoor zorgen dat sommige symbolen naar voren lijken te komen en zo onbedoeld meer nadruk krijgen. Dit kan voorkomen worden door de kleur en verzadigingsgraad goed te kiezen, door belangrijke objecten in hun geheel af te beelden en door de objecten genoeg ruimte te geven in het kaartbeeld zodat het beeld niet te druk wordt;

3.2.2 Kaartontwerp voor een internet GIS

Voor het ontwerpen van een kaart op het beeldscherm gelden andere regels dan voor het ontwerpen van een kaart op papier. De grootste beperkingen zijn een lage resolutie en een beperkte weergavegrootte. Daartegenover staat dat er gebruik gemaakt kan worden van allerlei interactieve tools zoals verschuiven, zoomen en extra informatie opvragen. Juist deze tools vragen om een goed basisontwerp, omdat ze kunnen zorgen voor een slecht overzicht en slechte (of onjuiste) kaarten. Bij (te veel) inzoomen kan bijvoorbeeld een kaartbeeld met overlappende gegevens ontstaan, waardoor de verschillende thema's niet meer onderscheidbaar zijn.

Voor internet GIS is dat nog wat ingewikkelder, omdat het gebruik van meerdere databronnen gevolgen heeft voor de kartografische regelgeving, doordat van tevoren niet bekend is welke data weergegeven moet worden. Bovendien is het onbekend of de verschillende datasets hetzelfde referentiesysteem kennen (bijvoorbeeld RD-coördinaten). Voordat de datasets samen afgebeeld moeten worden, moet het dus duidelijk zijn wat de coördinatenstelsels zijn van de afzonderlijke datasets, wat het gewenste coördinatenstelsel is en of de gekozen datasets getransformeerd kunnen worden indien nodig. Voor de uiteindelijke visualisatie is het belangrijkste dat de gegevens worden gepresenteerd in één coördinatenstelsel.

Kijken we nog eens naar de onderdelen van een internet GIS [zie §2.4][aangepast naar Rengelink, 1999]:

1. **het grafische model;**
2. **de gebruikers-interface**;
3. **de query-functionaliteiten**.

We zien dat vooral het eerste deel van het internet GIS het ontwerp van de kaart bevat, maar de twee andere onderdelen kunnen wel invloed uitoefenen op de regels door bijvoorbeeld de manier waarop een bepaalde regel geactiveerd wordt (zoals in- en uitzoomen, of het aan- of uitzetten van lagen).

Er zijn een aantal aandachtspunten in het grafische model te onderscheiden die in regels gevat zouden kunnen worden. Deze aandachtspunten zijn:

1 schaal en schaaldomein

welke schaal heeft het uit te beelden thema, en voor welke schaalbereik kan dit gebruikt worden;

2 generalisatie

hoe en op welke basis vindt generalisatie plaats (hangt samen met schaal en schaaldomein;

3 semantiek

wat is de betekenis van de thema's, hoe kan de meta-data uitgelegd en begrepen worden, hoe combineer je twee (dezelfde) lagen van verschillende bronnen;

4 plaatsen van tekst

Wat is de beste plek voor een tekst, kan deze in sommige gevallen weggelaten worden of vervangen worden door een pop-up tekst, moet de tekst een vaste plaats hebben, hoe wordt de tekst geschaald of geroteerd etc.;

5 belang van thema's

wat is de onderlinge verdeling als er meerdere thema's worden afgebeeld en hoe wordt deze onderlinge verdeling bepaald. Zijn er meerdere thema's tegelijk die allen een gelijke belangrijkheid hebben, is er een bijzondere focus op één thema en dient de rest als ondergrond, heeft de kaart een topografisch of een thematisch karakter etc.;

6 weergave/visualisatie

wat is een aanvaardbare hoeveelheid data (daaraan gerelateerd het aantal thema's, rekening houdend met de schaal), is er voldoende onderscheid in kleur en vorm, is het schaaldomein van het thema in overeenstemming met de weergave (inname van ruimte door objecten en relatieve belangrijkheid), is er een duidelijke ondergrond of referentie, zijn er voorkeuren van de bron wat betreft visualisatie, etc.;

7 voorschriften van de bron

In welk dataformaat wordt de data aangeleverd (in dit afstudeeronderzoek wordt aangenomen dat voor een multi-bron situatie de data in de vorm van een DLM inclusief visualisatievoorkeuren wordt aangeleverd door de gegevensbron) en hoeveel ruimte laat dat voor het zelf veranderen van weergave, wat zijn de visualisatievoorschriften die de bron meestuurt, kan daarvan afgeweken worden en zo ja, in welke mate, etc.;

8 gebruikers interface

Bij een internet GIS kan de gebruiker zelf invloed hebben op het kaartbeeld door het aanzetten van nieuwe thema's, highlighten etc. Ook overrulen van de kartografische regels en waarschuwingen van het systeem aan het adres van de gebruiker moeten als regels of acties worden gedefinieerd.

Aan de hand van deze aandachtspunten zullen regels opgesteld moeten worden die het automatiseren mogelijk maken.

3.2.3 Stappenplan voor kaartontwerp

De volgende stap is het onderkennen van verschillende fases van het ontwerpen van een (beeldscherm)kaart met een internet GIS en welke gegevens daarbij nodig zijn. [naar van der Schans, 1999]

1 Doel en functie van de kaart vaststellen (input gebruiker)

- functie en groep gebruikers;
- het gaat hier om weergave en bevraging;
- nauwkeurigheid van de data is vastgelegd in de bron (ook impliciet door de schaal);
- doel is niet altijd bekend!

2 Keuze van de thema's en verdeling van de thema's

- welke ruimtelijke elementen en thema's moeten worden weergegeven (input van de gebruiker);
- overzichtskaart – welke thema's blijven zichtbaar in de overzichtskaart;
- vaste (topografische) ondergrond;
- transparante overlays;
- titel van de kaart (wat zien we in de kaart; kan ook kaarttype of themanaam zijn);

- restricties aan thema's/combinaties van thema's zoals de bron dat heeft voorgeschreven. Opvolgen of overrulen?

3 Informatieanalyse van de thema's met behulp van meta-informatie

- welke attributen zijn er bij de thema's;
- onderlinge relatie tussen attributen;
- attributen opvraagbaar;
- nauwkeurigheid en betrouwbaarheid in tijd/geometrie/attributen.

4 Keuze grafische variabelen

- wat zijn de grafische elementen;
- onderscheidbaarheid van de grafische elementen;
- onderscheidbaarheid van overlappende elementen;
- waarnemingseigenschappen;
- expressieniveau van grafische expressievorm moet overeenkomen met het meetniveau van het weergegeven attribuut;
- gebruik van tekst;
- contrast tussen grafische expressie;
- relatie ruimtelijk object en attribuut (labellingprincipe);
- schaalverhouding;
- grafische generalisatie/meetkundige precisie;
- basiskaart mag niet te nadrukkelijk aanwezig zijn;
- lijndikten/raster/grootte wat gebeurt er met zoomen;
- interactieve aanpassing van de kaart;
- vormgeving is afgesteld op de functie van de kaart.

5 Legenda en gebruikers-interface

- moet volledig zijn;
- moet duidelijk leesbaar zijn;
- de structuur van de legenda moet relaties van de weergegeven attributen en attribuutwaarden duidelijk weergeven;
- voor ieder thema moet er een bronverwijzing komen;
- legenda en kaart moeten tegelijkertijd in beeld zijn;
- expressievormen van de kaart en de legenda moeten exact overeenkomen;
- er moet een referentie zijn met de naam van het gebied, of dat moet in de vorm van een coördinaatraster;
- noordpijl of coördinatenraster;
- schaalaanduiding die meeschaalt;
- bronvermelding per thema in de legenda;
- dialoogschermen die eventuele fouten kunnen voorkomen waarbij de input van de gebruiker wordt gevraagd als het systeem op grond van de informatie die hij heeft geen beslissing kan nemen, of als de gebruiker tegen deze regels in andere dingen wil doen;
- highlights van de thema's die de gebruiker interessant vindt, eventueel mouse-over highlights.

Bij het ontwerpen van regels dient dit als een richtlijn voor het proces wat een kartograaf doorloopt om tot een uiteindelijke ontwerpkeuze te komen.

3.3 Uitwerking per onderwerp

Het is nu duidelijk wat de stappen zijn die doorlopen moeten worden om een kaart te ontwerpen en welke thema's hierbij belangrijk zijn. Nu wordt het proces van het kaartontwerp verder uitgewerkt in wat specifiekere regels per onderwerp zoals deze in § 3.2.2 beschreven zijn.

3.3.1 Algemene aanpak

Voor een beeldschermkaart op het internet is het goed om een geografische referentie te hebben, bijvoorbeeld een topografische ondergrond. Als deze puur als referentie dient (en dus als achtergrond) kan deze het best in grijstinten of een kleur met lage verzadigingsgraad weergegeven worden, eventueel semi-transparant als de ondergrond boven een ander thema geplaatst wordt. De verschillende thema's kunnen dan in contrasterende kleuren weergegeven worden. Indien onderdelen uit deze geografische referentie een voorgrond-thema zijn worden deze niet in grijs weergegeven.

De kaartlagen worden gevormd door thema's die uit verschillende bronbestanden door de gebruiker geselecteerd worden. Deze thema's zijn in de legenda terug te vinden. De bronbestanden kennen vaak al een onderverdeling in verschillende soorten uit dat thema, wat direct een indicatie geeft over de hiërarchische structuur van het thema en de uiteindelijke weergave. Onderscheid moet gemaakt worden in topografische thema's die meestal naar belangrijkheid (een rijksweg of doorgaande weg is voor het kaartbeeld doorgaans belangrijker dan andere wegen) en naar geschiktheid (kleine objecten zijn niet geschikt voor kleinschalige kaarten) voor de schaal worden afgebeeld, en thematische thema's, die meestal naar oppervlakte en naar gewenstheid in de kaart worden afgebeeld. Een belangrijke reden voor het afbeelden van topografische objecten naar belangrijkheid en geschiktheid is dat vele topografische objecten ook daadwerkelijk als object in de werkelijkheid te vinden zijn, bijvoorbeeld spoorwegen. Het afbeelden van grondgebruik zoals grasland is alleen voor sommige toepassingen belangrijk [van der Schans, 1999].
Voor ieder thema moet worden vastgesteld voor welke minimale en maximale schaal de gegevens nog gebruikt kunnen worden, eventueel met generalisatie. Ook moet de schaal altijd op het beeldscherm weergegeven worden liefst met een schaalbalk die met de kaart meeschaalt. Hoewel er altijd een schaal is (de afmetingen van objecten op het beeldscherm hebben altijd een relatie met de afmetingen van objecten in de werkelijkheid) is deze niet altijd vast precies vast te stellen, omdat de grootte van het beeldscherm niet precies bekend is. Het is daarom nuttig de inhoud van het beeldscherm te koppelen aan een bepaald printformaat. Als bekend is dat het beeldscherm op bijvoorbeeld A3 wordt afgedrukt, kun je de schaal bepalen.
In principe kunnen sommige thema's voor uiteenlopende schalen bedoeld zijn. Een hoofdweg moet bijvoorbeeld op zowel klein- als grootschalige kaarten aanwezig zijn.
Conventies en internationale afspraken over kleuren zijn belangrijk: water is blauw, bos is groen, huizen zijn rood. Andere onderdelen kunnen variëren, maar wel met één kleur per onderdeel of thema. Ook verschillende entiteiten kunnen in sommige gevallen wel dezelfde kleur hebben. In het algemeen geldt: geen verwarring scheppen, bij twijfel niet doen [Rengelink, 1999].
De verschillende thema's hebben attributen die nooit allemaal in het beeldscherm verwerkt kunnen worden. Afhankelijk van het attribuut wordt het in het beeldscherm geplaatst (meestal als tekst) of als attribuuttabel aan het object gehangen. Dat object moet dan wel

pickable of clickable zijn, zo mogelijk met een highlight aangegeven. Highlight gebruikt vaak heldere kleuren (zoals geel), maar pas op dat originele kleuren (in de legenda) niet onduidelijk worden. Het selecteren van grote polygonen of veel kleinere polygonen tegelijk kan tot gevolg hebben dat zeer grote gebieden een (gekleurde) highlight krijgen. Eén oplossing is een transparante highlight, een andere oplossing is het highlighten van de randen van deze gebieden, met als nadeel dat deze randen niet in het kaartbeeld te zien zijn.
Bij rasterbestanden moet bekeken worden of:

- het rasterthema onderop geplaatst kan worden;
- bij meerdere rasterthema's (semi)transparante weergave;

Het is gemakkelijker om meerdere vectorbestanden te gebruiken dan meerdere rasterbestanden. De logische hiërarchische structuur van de geometrische entiteiten is onder op vlaktypen, daarboven lijntypen, daarboven punttypen.

3.3.2 Schaal

De gebruiker bepaalt de (initiële) weergaveschaal door een bounding box of interessegebied op te geven in bijvoorbeeld een overzichtskaart. De gegevens die daarna opgevraagd worden moeten in dat gebied liggen en ook op die schaal weergegeven kunnen worden. Per thema moet dus aangegeven worden welk *gebied* die data beslaat en op welke *schaal* die data afgebeeld kan en mag worden. Een thema met schaal 1:10.000 kan dan bijvoorbeeld in de schaalbereik van 1:15.000 tot 1:5000 afgebeeld worden. Bij in- of uitzoomen moet bekeken worden of het thema met de nieuwe schaal weergegeven kan worden.
Als de gewenste schaal van de gebruiker veel groter of kleiner is dan die van het gekozen thema, moet afgewogen worden of het thema wel of niet geschikt is voor deze andere schaal. Er moet in dat geval gelet worden op hoeveelheid data per oppervlakte-eenheid op het beeldscherm, of de (vector)data een punt-, lijn- of vlakgeometrie. Bij rasterdata moet gekeken worden of de resolutie van het af te beelden thema wel voldoende is bij de gewenste weergaveschaal.

Als de bron voorschriften meezendt over het simultaan gebruiken van verschillende thema's (dat wil zeggen een verplichte koppeling van twee of meer thema's) kan het gebeuren dat het ene thema een grotere schaalbereik heeft dan het andere thema, terwijl de bron wel gebiedt dat ze zonder elkaar niet afgebeeld mogen worden. In zo'n geval beeld je de thema's af in de schaal waarin ze afgebeeld kunnen worden, dus de schaalbereik die de bron aangeeft, maar moet de gebruiker ingelicht worden over het ontbreken van een thema in de huidige schaal, en dat deze door in te zoomen wel afgebeeld kunnen worden.
Een dergelijke oplossing kan ook gebruikt worden als de verschillende thema's uit verschillende bronnen komen en wat schaal betreft niet samenvallen. Bij bepaalde weergaveschaal moet het systeem dat dan wel aangeven dat sommige thema's niet zichtbaar zijn. Probeer dus het systeem een schaal te laten bepalen die de wensen van de gebruiker het meest benaderd, en waarop zoveel mogelijk thema's te zien zijn.

Bij het afbeelden van een thema in de buurt van de minimale of maximale schaal kan het voorkomen dat er te veel of juist te weinig data in het beeld komt. Dit wordt soms opgelost als een vorm van generalisatie, bijvoorbeeld bij steden: Bij kleinschalige kaarten is de stad vaak gereduceerd tot een stip, bij een middelgrote schaal wordt de stad veranderd in een werkelijke outline van de stadsgrenzen (oranje vlak), bij een grote schaal worden de afzonderlijke bouwblokken zichtbaar met straten.

De vraag echter hoe je bepaalt wanneer er teveel of te weinig data te zien is. Het gemakkelijkst lijkt een bepaalde range van aantal datapunten per oppervlakte-eenheid (bijvoorbeeld 10 bij 10 pixels) vast te stellen en zo te veel of te weinig data kunnen constateren. Of er werkelijk te veel of te weinig data in het beeld te zien is, is echter sterk afhankelijk van de thema's en de combinatie daarvan. Een beslissing "te veel data" door het systeem aangegeven kan op het beeldscherm echter nog overzichtelijk overkomen.
Ook kan er plaatselijk veel data zijn, terwijl er over het hele beeld misschien wel erg weinig data is. Dit kan bijvoorbeeld voorkomen bij een combinatie van meerdere thema's met polylijnen zoals de bekabeling van een nutsbedrijf, de randen van de wegen etc. De concentratie van lijnen zal dan langs de wegen groter zijn dan de concentratie van lijnen ver buiten die wegen. In het algemeen kan echter gesteld worden dat een combinatie van thema's met verschillende geometrie zorgt voor een evenwichtiger beeld dan een combinatie van thema's met dezelfde geometrie.

3.3.3 Generalisatie

Generalisatie is één van de moeilijkste onderwerpen als het gaat om het automatiseren van geoinformatieverwerking, maar het is tegelijkertijd één van de meest fundamentele onderwerpen in het kaartontwerpproces. Uit de beschikbare informatie moeten de essentiële onderdelen geselecteerd worden en die moeten op zo'n manier weergegeven worden dat het resultaat zowel duidelijk als informatief is. De belangrijkste factoren voor generalisatie zijn de schaal, omdat deze de beschikbare ruimte beperkt, en het doel van de gebruiker. Kaartgeneralisatie kan dus gezien worden als een schaalafhankelijk proces van informatie-abstractie ten behoeve van het beoogde doel van de gebruiker.

Er zijn twee soorten generalisatie [Jones, 1997]:

1 **semantische generalisatie**: generalisatie vindt plaats op basis van de betekenis en de functie van de kaart en dit proces is afhankelijk van het al of niet aanwezig zijn van hiërarchische structuur in de geografische informatie. De belangrijkste hiërarchische structuren zijn classificatie (hydrografie, infrastructuur, stedelijke gebieden) en aggregatie (een stad bestaat uit stadsdelen, administratieve wijken, wijken, buurten etc.). Voor het automatiseren van deze vorm van generalisatie moet het systeem dus deze hiërarchische structuren kunnen herkennen, ze begrijpen, en ze toe kunnen passen op de gewenste schaal. In het DLM- DKM model komt deze vorm van generaliseren neer op een kleine aanpassing van het DLM, waarbij een 'nieuw' of tijdelijk DLM ontstaat;
2 **geometrische generalisatie**: De grafische representatie van de data vereist symbolisatie van de geselecteerde informatie en de keuze van grafische elementen en eventueel tekst. Geometrische generalisatie vindt plaats op basis van deze grafische representatie en wordt min of meer bepaald door het samenspel van semantische generalisatie, de symboolkeuze en de beperkingen van de kaartschaal. De bewerkingen die bij deze soort generalisatie aan de orde komen zijn selectie, vereenvoudiging, vergroting, verplaatsing, samenvoeging, symbolisatie en benadrukking/verwaarlozing. In het DLM - DKM model komt deze vorm van generaliseren neer op het wijzigen van het oorspronkelijke DKM naar een 'nieuw' of tijdelijk DKM.

Dit onderscheid in generalisatie wordt ook wel respectievelijk conceptuele en grafische generalisatie genoemd.

Een groot probleem bij de eerste vorm van generalisatie is dat de semantiek niet altijd aan een bepaalde standaard voldoet. Bij een multi-bron internet GIS hebben verschillende

bronnen verschillende namen voor objecten die op hetzelfde neerkomen, bijvoorbeeld codes in plaats van namen. Er is dan veel extra informatie nodig om te begrijpen wat de data precies voorstelt. Vanwege de complexiteit van het onderwerp heeft het de voorkeur om generalisatie over meerdere bronnen zoveel mogelijk te vermijden. Meer over semantische aspecten in § 3.3.4.
De tweede vorm van generalisatie is minder gebonden aan semantiek en is beter geschikt om regels over op te stellen. Het is wel van belang te weten welke thema's belangrijk zijn en welke wat minder.

De randvoorwaarden van het generalisatieproces zijn:

- het doel van de kaart (generaliseren betekent vaak informatieverlies);
- de schaal en schaaldomein van de kaart (hoe kleiner de schaal, hoe minder ruimte er beschikbaar is voor de gegevens);
- de grafische beperkingen (minimale lijndikte, onderscheidend vermogen van het menselijk oog);
- de aard van de te generaliseren informatie (kwalitatieve of kwantitatieve informatie).

Deze factoren bepalen voor een groot deel de manier waarop gegeneraliseerd wordt en zijn van grote invloed op het eindresultaat. Overigens is het belangrijk te onderkennen dat dezelfde randvoorwaarden niet altijd een uniform resultaat opleveren, omdat een gedeelte van het generalisatieproces afhankelijk is van menselijke interpretatie en handelen.

Omdat er bij internet GIS met verschillende thema's gewerkt wordt, moeten de randvoorwaarden van de verschillende thema's samengevoegd worden en vergeleken worden. Het is duidelijk dat er soms tegenstrijdige voorwaarden voor kunnen komen, bovendien kan het voorkomen dat in een bepaald thema generalisatie niet wenselijk is (bijvoorbeeld voor het afbeelden van eigendomspercelen). Ook tekst in het kaartbeeld is onderhevig aan generalisatie omdat de hoeveelheid tekst per eenheid oppervlakte al snel te groot wordt. Meer over tekstplaatsing in § 3.3.5.

Automatisering van generalisatie is voor (interactieve) beeldschermkaarten nog lastiger dan bij statische beeldschermkaarten omdat de schaal van de kaart kan veranderen en de voorwaarden dus telkens veranderen. Doorgaans zal generalisatie voorkomen als er wordt gezoomd zodat de kaart een andere schaal krijgt. De bronbestanden hebben een schaalbereik die aangeeft voor welke schaal ze nog geschikt zijn, maar vaak moet er wel het een en ander aangepast worden in de weergave. Hier zijn twee factoren van belang:

1 **kwantitatieve factoren**: schaalverschil en daarbij behorende ruimte die objecten in kunnen nemen;
2 **kwalitatieve factoren**: relatieve belangrijkheid, wat pas je aan en wat niet, wat geef je voorrang als er een keuze gemaakt moet worden.

Problemen bij [1] zijn meestal makkelijk aan te passen door bijvoorbeeld de lijndikte te veranderen. Problemen bij [2] zijn afhankelijk van het doel van de kaart, daarom zijn er geen algemene regels die gevolgd kunnen worden. Het moet echter mogelijk zijn de gebruiker aan te laten geven wat zijn prioriteiten zijn.

Net als bij de bepaling van de schaal is het bij generalisatie van belang om te weten of het kaartbeeld te vol dreigt te worden en generalisatie noodzakelijk is voor een evenwichtig kaartbeeld. Het weglaten van één of meerdere thema's kan in dit opzicht ook als generalisatie aangemerkt worden. Er zijn verschillende oplossingen te noemen als het kaartbeeld te vol dreigt te worden:

- Thema's weglaten op basis van regels over welke thema's in dit kaartbeeld de beste ondergrond/ondersteuning bieden. Hier komen de semantische aspecten aan de orde: welke thema's zijn dit, wat wil de gebruiker graag als ondergrond zien en welke naam geeft de bron aan deze thema's;
- Thema's weglaten op basis van de prioriteiten van de gebruiker De hiërarchie van de thema's worden aangegeven door de gebruiker: de eerste keus heeft de hoogste prioriteit, de laatste keus heeft de laagste prioriteit. Deze hiërarchie kan in de legenda weergeven worden en eventueel makkelijk veranderd kunnen worden. Een gevaar bij deze oplossing is dat de gebruiker zo een kartografisch niet-verantwoorde kaart samenstelt en de informatie-overdracht niet correct is;
- Alle thema's laten staan met een waarschuwing over de gevaren van een te volle kaart.

Ondanks de vele problemen die het automatiseren van generalisatie met zich meebrengt zijn er zeker mogelijkheden op het gebied van generalisatie van lijn- en vlaksymbolen, zoals het weglaten van punten uit het coördinatenreeks. Om de karakteristieken van de lijn wel te bewaren kan bijvoorbeeld het Douglas/Puecker lijnvereenvoudigingsalgoritme gebruikt worden [Jones,1997]. Omdat generalisatie een onderwerp op zich is wordt in dit onderzoek verder niet op generalisatie ingegaan.

3.3.4 Semantiek

Om gegevens te kunnen gebruiken en manipuleren is het noodzakelijk de betekenis, of de semantiek van de data te kennen. De betekenis van objecten wordt vaak toegekend in de vorm van classificaties met een subset categorieën (meta-data), die de fenomenen in de werkelijke wereld beschrijven in verschillende abstractieniveaus.
Maar wat is de betekenis van de objecten in de werkelijke wereld (wat is een formele beschrijving van een weg, een gebouw etc.) en in hoeverre is de naamgeving van de gegevens gestandaardiseerd? Bij het werken met meerdere bronnen hanteert iedere bron meestal zijn eigen semantiek, terwijl niet altijd duidelijk is wat het voorstelt en of objecten uit verschillende bronnen misschien dezelfde objecten zijn met een andere naam. De objectklasse die in de ene database "bouwblokken" heet en in de andere database "woningen" heet kan samen in een gecombineerde weergave als één objectklasse worden weergegeven. Het systeem moet dan echter over beslisregels kunnen beschikken die dit mogelijk maken. Het is dan noodzakelijk dat alle mogelijke objectklassen en objectnamen in het systeem ingevoerd worden en dat het systeem weet welke objectklassen samengevoegd kunnen worden. De bronvermelding van de verschillende objecten moet echter wel behouden blijven, maar wat als twee objecten net niet overlappen? Het is duidelijk dat samenvoegen van objectklassen op deze manier een onbegonnen zaak is omdat de semantiek van de verschillende bronnen te heterogeen is, bovendien zullen er altijd kleine verschillen bestaan tussen dezelfde objecten in de verschillende bronnen omdat woorden vaak niet eenduidig zijn.

Bij het gebruik van meerdere bronnen is er zo'n veelheid en verscheidenheid aan objectklassen dat het onmogelijk is om al die objectklassen te 'begrijpen' zonder één of andere vorm van standaardisatie. In Nederland is er al een soort van standaard in de vorm van een Terreinmodel Vastgoed: typografie, leidingen, gemeenten etc. Het wordt lastiger als je bronnen gaat gebruiken die deze standaard niet gebruiken.
In dit onderzoek wordt niet verder ingegaan op de semantiek van de gegevens en worden alle gegevens als 'begrepen' behandeld.

3.3.5 Plaatsen van tekst

Het plaatsen van tekst in het kaartbeeld is lastig omdat er niet één juiste lokatie is (de tekst zelf heeft geen geografische component). Dat geeft je wel veel keuzevrijheid, want het is makkelijk te verplaatsen als het in de weg staat. Belangrijk bij het plaatsen van tekst is dat het duidelijk blijft bij welk geografisch object het tekstlabel hoort en dat het niet (te veel) overlapt met andere geografische objecten en de bijbehorende tekstlabels. Een nieuwe mogelijkheid bij beeldschermkaarten is het gebruik van pop-up windows, dus de tekst komt tevoorschijn als er met de muis overheen bewogen wordt. Alleen de belangrijkste aanknopingspunten hoeven dan standaard in de kaart weergegeven worden, de rest van de tekst blijft voor een belangrijk deel onzichtbaar maar wel aanwezig.

Omdat het (handmatig) plaatsen van tekst veel tijd in beslag neemt en afhankelijk is van de gekozen thema's en de weergaveschaal, is automatisering van het plaatsen van tekstlabels zeer wenselijk en ook goed mogelijk, mits er een goede methode wordt gebruikt om de juiste plaats te bepalen.
Automatisering van tekstlabeling kan bijvoorbeeld via de volgende stappen [Jones, 1997]:

1. specificatie van de kaartonderdelen (features en tekstkarakteristieken);
2. tekstfont en grootte bepalen (per type feature één font en grootte);
3. genereren van proef-tekstposities (in de orde van voorkeur);
4. selectie van optimale labelposities;
5. relatieve prioriteit van kaartfeatures vaststellen om op grond hiervan positie en overlap te bepalen;
6. features met tekstrangorde toebedelen.

Ook het plaatsen van tekst is zeer complex en wordt in dit onderzoek verder niet behandeld.

3.3.6 Belang van thema's

Omdat er bij internet GIS meerdere thema's uit meerdere bronnen worden gecombineerd, is het belangrijk vast te stellen wat de onderlinge belangrijkheid van de thema's zijn. Bij een enkele bron is de belangrijkheid van de thema's vaak min of meer gelijk omdat ze meestal een gezamenlijke functie hebben, of de belangrijkheid ligt in de aard van de gegevens en de bijbehorende classificatie. Als deze thema's los van de bron komen, en er meerdere thema's worden gecombineerd, volgt daar niet zomaar uit welk thema de gebruiker het belangrijkst vindt en welke hij enkel als referentie wil gebruiken. Het ligt voor de hand dat de belangrijkheid van het thema aangegeven wordt door de gebruiker door middel van prioriteiten. Deze prioriteiten zijn ook van belang voor het generaliseren (weglaten, weergave wijzigen) van de thema's.
Deze prioriteiten kunnen op meerdere manieren vastgesteld worden:

1. De volgorde van belangrijkheid wordt vastgesteld met behulp van de volgorde waarin de thema's worden opgevraagd: wat het eerst opgevraagd wordt zal wel het belangrijkste onderdeel zijn (met uitzondering van de ondergrond). Dit is een nogal starre methode die veel vooronderstellingen kent;
2. De gebruiker moet, na het opvragen van de thema's die hij wil zien, aangeven welke thema's hij het belangrijkst vindt. Hij kan de thema's een rangorde geven (nummeren) en daar eventueel ook gewichten aan geven. Bij het opvragen van een nieuw thema wordt telkens bepaald welk gewicht de gebruiker toe wil kennen aan het nieuwe thema. Dit kan bijvoorbeeld door aan ieder thema een nummer toe te kennen waarbij het laagste

nummer de hoogste prioriteit kent. De gebruiker moet op deze manier wel veel extra werk doen wat niet de bedoeling is;
3. De legenda laat de thema's zien in volgorde van opvragen (het eerste thema staat bovenaan, het laatste onderaan) en de gebruiker kan de volgorde van de thema's veranderen door te slepen in de legenda. Op deze manier kun je ook de belangrijkheid continu veranderen. Deze oplossing lijkt de beste van de drie, omdat de gebruiker zonder al te veel extra werk zelf invloed kan uitoefenen op de prioriteit van thema's.

Het kan natuurlijk gebeuren dat er meerdere thema's zijn die allen even belangrijk zijn of geen onderscheid hebben. In dat geval is het niet erg dat de thema's toch een rangorde hebben, zolang alle thema's maar aan kunnen blijven staan. Pas op het moment dat het kaartbeeld te vol wordt en er thema's uitgezet of van weergave veranderd moeten worden, wordt besloten welk thema dat moet worden.

3.3.7 Weergave en visualisatie

Bij het visualiseren van gegevens op een beeldscherm dient er rekening gehouden te worden met een beperkte resolutie en grootte van het afbeeldingsgebied. Dit laatste hoeft natuurlijk geen probleem te zijn omdat de gebruiker het afbeeldingswindow kan verschuiven naar het interessegebied. Het is dan wel belangrijk dat de gebruiker een goed overzicht houdt over waar hij zich bevindt. Wat betreft de resolutie van het beeldscherm, is het belangrijk dat de gekozen grafische weergave in die resolutie ook overkomt. Ook het gebruik van kleur is belangrijk, omdat de kleuren op een beeldscherm er anders uitzien dan op een papieren kaart (zie § 2.1.2). Het visualiseren op een beeldscherm zorgt ook voor nieuwe mogelijkheden. Naast de al genoemde mogelijkheden tot het aanroepen van achterliggende informatie door middel van pop-up windows is in dit geval transparantie een middel om objecten een kleur te geven, terwijl er dan geen 'dode' vlakken ontstaan: onzichtbare gebieden onder vlakken die gevuld zijn. Daarnaast kan transparantie ervoor zorgen dat de onderliggende informatie nog steeds zichtbaar is, er kunnen dan ongewenst wel mengkleuren ontstaan.

Voor het visualiseren van gegevens is het belangrijk de regels over de visuele variabelen [Bertin, zie ook §2.1.1] in acht te nemen. In de gewenste toekomstige situatie van een internet GIS worden echter niet alleen gegevens verstuurd, maar ook een bepaalde voorkeursvisualisatie die de bron toegekend heeft aan de gegevens. De verschillende thema's die uit verschillende bronnen door gebruikers worden opgevraagd, worden iedere keer in verschillende combinaties gepresenteerd. Het heeft dan de voorkeur de weergave per thema uniform trachten te houden. Dit is ook van belang als een gebruiker een bepaalde combinatie opnieuw opvraagt. Het is dan beter als de tweede kaart er hetzelfde uit zou zien als de eerste kaart. Daarnaast is het bij het gebruik van meerdere bronnen belangrijk dat er per bron zo mogelijk een uniform product afgeleverd wordt. Het ligt voor de hand dat de bron, naast de gegevens en de meta-gegevens, informatie meestuurt met een voorkeursweergave (bv. kleur, lijndikte en -stijl) voor de betreffende gegevens. Het systeem kan hier dan zijn manier van weergeven op baseren. Deze informatie kan bijvoorbeeld in de vorm van een XSLT (stylesheet) zijn [zie ook § 2.3.5]. Hoewel de weergave door het systeem bepaald wordt, biedt het ook de mogelijkheid aan de gebruiker de standaard weergave te veranderen.

Bij het combineren van thema's uit verschillende bronnen, waarbij de thema's allemaal hun eigen weergavevoorkeur hebben, is de kans groot dat dit conflicten oplevert. Het systeem

speelt een grote rol bij het oplossen van die problemen, maar heeft wel de feedback van de gebruiker nodig. Als de gebruiker bijvoorbeeld twee thema's uit twee bronnen heeft geselecteerd die beiden voorkeurskleur "rood" hebben, moet er een keuze gemaakt worden welk thema deze kleur kan behouden. De prioriteiten die de gebruiker aan de verschillende thema's geeft is dan doorslaggevend. Het kan natuurlijk voorkomen dat beide thema's onder een zelfde klassificatie vallen, bijvoorbeeld "gebouwen", en dus eigenlijk samengevoegd kunnen worden en dus beiden de voorkeursweergave kunnen behouden. Het is echter moeilijk te bepalen of de thema's misschien in grote lijnen hetzelfde zijn, maar verschillen in bijvoorbeeld nauwkeurigheid of actualiteit. Omdat dit een moeilijk semantisch probleem is, wordt hier verder niet op ingegaan, maar dit zou in theorie een oplossing kunnen bieden. Doordat de eerste voorkeursweergave niet altijd opgevolgd kan worden, moet er dus ook een tweede en mogelijk een derde voorkeursweergave worden opgesteld.
Ook binnen één bron kunnen er voorschriften worden verbonden aan het weergeven van thema's, bijvoorbeeld als een thema gerelateerd is aan een ander thema en de weergave van de één afhankelijk is van de ander. Als er bijvoorbeeld drie thema's (thema1a, thema2a en thema4a) van bron A worden gecombineerd met twee thema's van bron B, en er zijn voorwaarden voor meer dan deze drie thema's uit bron A, zoals dat thema 1a niet zonder thema 5a afgebeeld mag worden, moeten deze losse thema's aangepast worden zodat misschien ook thema 5a in het totaalbeeld komt. Losse thema's (dus thema's die niet aan andere thema's gebonden zijn) kunnen dus probleemloos aangepast worden, bij gerelateerde thema's is het belangrijk te kijken wat de bron daar precies over zegt en hoe je dat nog aan kan passen, of eventueel maar niet.

De vraag is nu tot hoe ver de gebruikersvrijheid mag gaan als het gaat om veranderen van weergavevoorschriften en keuzes die het systeem heeft gemaakt. Het internet GIS is aan de ene kant bedoeld om de gebruiker zelf te laten bepalen wat hij ziet en hoe hij dat ziet, maar aan de andere kant kan dit niet betekenen dat de kartografische verantwoording geheel links komt te liggen. Er zijn verschillende oplossingsscenario's:

1. de gebruiker mag niets veranderen, het kaartbeeld blijft hetzelfde;
2. de gebruiker mag sommige onderdelen veranderen, het systeem geeft een waarschuwing op het moment dat veranderingen leiden tot een onevenwichtig kaartbeeld;
3. de gebruiker mag alle onderdelen veranderen, het systeem geeft een waarschuwing op het moment dat veranderingen leiden tot een onevenwichtig kaartbeeld.

Voor een multi-bron GIS met onbekende gebruikers en doelen lijkt de laatste oplossing de meest geschikte.

3.3.8 Voorschriften bron

De bruikbaarheid van een kartografisch expertsysteem hangt sterk af van de bronnen die de bestanden sturen. De eerste vraag is dus hoe de data wordt aangeleverd. In §2.3.4 waren deze drie mogelijke vormen uiteengezet:

Picture case
Op aanvraag van de gebruiker wordt er door de webserver een plaatje van de kaart, meestal in de vorm van een GIF of JPEG. Het plaatje is gemaakt door de MapServer. Het voordeel van deze methode is dat de visualisatie per kaartje goed is. Een nadeel is echter dat er verder

niets meer aan veranderd kan worden, en dat het nagenoeg onmogelijk is om gegevens uit verschillende bronnen te combineren.

Graphic Element Case
De Webserver stuurt een verpakte set van individuele grafische elementen in een bepaalde projectie en referentiesysteem. Sommige grafische elementen zijn zelf een vooraf gemaakt plaatje, dus de picture case kan een subset zijn van de Graphic Element Case. De weergave van de grafische elementen kan (in beperkte mate) aangepast worden, maar het gevaar daarbij is dat de gebruiker de semantiek van de grafische elementen niet kent. Omdat het onduidelijk is wat de grafische elementen voorstellen, is er geen enkele grond waarop men kan beslissen of een grafisch element aangepast kan worden. Ook bij het combineren met verschillende bronnen is het door het ontbreken van semantiek onduidelijk of er conflicten zijn.

Data case
In deze vorm wordt de data zelf van de server naar de gebruiker verzonden, inclusief meta-data en visualisatievoorschriften. Met behulp van deze additionele informatie is de gebruiker in staat de verschillende typen gegevens te onderscheiden (ofwel de gebruiker kent de semantiek) en heeft hij ook een grond voor het maken van keuzes.
Het is belangrijk dat de voorschriften er zijn, omdat de keuzevrijheid anders zo groot wordt dat dit kan leiden tot een kartografisch onjuiste visualisatie. Het moet echter mogelijk blijven de voorschriften te negeren en de visualisatie aanpassen aan de wensen van de gebruiker.

Aangezien de eerste en tweede vorm van data-overdracht in dit afstudeeronderzoek niet relevant zijn, omdat de browser of GIS-software van de eindgebruiker niet veel in kan brengen in de visualisatie, gaan we er dus van uit dat voor het goed kunnen gebruiken van een kartografisch expertsysteem de data naar de gebruiker wordt verstuurd, inclusief meta-informatie en visualisatievoorschriften (data-case).
In de data-case kunnen de gegevens in twee gegevensstructuren opgestuurd worden:

RASTER
een rasterbestand is meestal beeldvullend, dat wil zeggen dat de onderliggende informatie niet meer te zien is als de rasterlaag weergegeven is (behalve als de lagen semi-transparant weergegeven worden). In een gecombineerde weergave is het dus het beste als er maar één rasterbestand per visualisatie te zien is, en dat deze onder alle andere thema's ligt. Een mogelijke oplossing voor deze beperking is het toevoegen van transparantie in de lagen. De bron moet dan wel de mogelijkheid bieden om deze transparantie aan te brengen. Bij het weergeven van bijvoorbeeld een satellietfoto als rasterbestand, is het belangrijk dat de vectordata van bijvoorbeeld de huizen en de wegen die op dit rasterbestand te zien zijn, goed aansluiten. Het belangrijkst blijft echter dat alle informatie die gevisualiseerd wordt, ook goed te zien blijft. Manipulatie en verandering in de weergave data is bij rasterbestanden zeer beperkt.

VECTOR
Het is aan te nemen dat er bij het combineren van data uit verschillende bronnen het accent ligt op het combineren van vectordata, en eventueel rasterdata als ondergrond. Juist bij vectordata kan er veel informatie gegeven worden over hoe de gegevens gevisualiseerd kunnen worden. Deze informatie kan bijvoorbeeld zijn:

- entiteiten of basisgeometrie
  - punt (met of zonder afmetingen);
  - lijn (polylijn);
  - vlak (polygoon);
- topologie
  - vlak bestaat uit verwijzing naar randen;
  - rand bestaat uit verwijzingen naar punten;
- referentiesysteem / coördinatenstelsel
- attributen
  - attribuuttype (semantiek): bepaalt o.a. kleur(klassen), gaat in op conventies;
  - attribuutwaarde: bepaalt grafische variabele of weergaveklasse;
- schaalbereik (minimale en maximale weergaveschaal, precisie van vectorbestanden);
- belangrijkheid (ingebracht door gebruiker);
- symboolkeuze;
- tijdstip van geldigheid.

Alleen schaalbereik en symboolkeuze (en in minder mate de attribuuttypen en -waarden) zijn van belang voor de totstandkoming van het DKM, de rest is onderdeel van het DLM.

Naast de informatie over de thema's afzonderlijk kan de bron ook voorschriften geven over het gebruik van de thema's en eventuele restricties daaraan (zie ook § 3.3.7). De bron kan bijvoorbeeld bepaalde thema's relateren en er de voorkeur aangeven dat de betreffende thema's niet afzonderlijk worden afgebeeld. Ook de wijze waarop de thema's afgebeeld worden kan gerelateerd zijn: indien thema A1 in rood wordt afgebeeld, moet thema A2 in bruin afgebeeld worden. Wordt de kleur van thema A1 veranderd in oranje, dan moet thema A2 veranderen in paars.

3.3.9 Gebruikers interface

Bij een internet GIS kan de gebruiker zelf invloed hebben op het kaartbeeld door het aanzetten van nieuwe thema’s, het veranderen van de weergave van de thema's, selectie op attribuutwaarde etc. Het komt er vaak op neer dat de gebruiker de voorschriften van de bron overschrijdt, echter zonder dat hij op de hoogte is van deze voorschriften. Het gevaar bestaat dat door deze veranderingen de kaart niet meer kartografisch verantwoord is. De vraag is nu welke vrijheden we de gebruiker toe willen kennen. De doelstelling van een KES is dat iedereen, ongeacht zijn of haar kartografische kennis, gebruik kan maken van de gegevens die via het internet beschikbaar worden gesteld. Een dergelijke opzet vraagt dus om een vorm van begeleiding en advisering van de gebruiker om de kartografische aspecten van het visualiseren van gegevens in goede banen te leiden.

Gebruikers die geen of weinig verstand hebben van kaarten maken zullen dus minder vrijheden ondervinden dan expert gebruikers. Voor beide soorten gebruikers is het echter wel belangrijk dat het systeem aangeeft dat iets niet mag of dat iets af te raden is. Deze waarschuwingen moeten ook als regels in het expertsysteem ingevoerd worden.

3.4 Regels voor het expertsysteem

Om de regels op te stellen die het systeem gaat gebruiken om de gegevens tot een juiste visualisatie te brengen, is het belangrijk de stappen van het kaartontwerpproces, zoals die in § 3.2.3 uiteengezet zijn, te omschrijven in stappen die het systeem neemt. Ook de mogelijke feedback van de gebruiker moet hierbij aangegeven worden.

Omdat dit onderzoek niet gericht is op een totale oplossing van het probleem, worden een aantal uitgangspunten geformuleerd, waarvan aangenomen wordt dat deze feitelijk waar zijn en dat het geheel goed werkt. Deze uitgangspunten zijn:

- bron stuurt voorschriften of hints over de gewenste weergave met de data mee;
- het systeem kent de meta-data (bijv. kaartlaagnamen en attribuutnamen);
- de server levert als in data-case: data + meta-data + voorschriften;
- regels over generalisatie, tekstplaatsing en semantiek worden niet behandeld.

Hoewel sommige regels voor de mens op het eerste gezicht eenvoudig lijken, kunnen deze zeer complex en daarom praktisch vrijwel onuitvoerbaar zijn, omdat het beslisproces wat er aan te pas komt te complex is om te implementeren in een geautomatiseerd systeem.

3.4.1 Regels in het stappenplan van het kaartontwerpproces

1 Doel en functie vaststellen

- gebruiker geeft door middel van geografische interactie het interessegebied op: gewenst referentiesysteem, coördinaten, bounding box, gebiedsnaam of andere naam gerelateerd;
- gebruiker geeft impliciet het doel door het stellen van prioriteiten.

2 Keuze thema's en verdeling thema's

- het systeem geeft een lijst van bronnen met de beschikbare thema's (of kaartlagen), de gebruiker kiest en de browser haalt de gegevens op;
- het systeem geeft aan dat de volgorde van kiezen de prioriteit aangeeft, hij geeft ook aan dat dit later veranderd kan worden (aangeven gebeurt in handleiding of on-line helpfunctie);
- het systeem bepaalt wat er in het overzichtsgebied komt, dit kan door de gebruiker veranderd worden, zowel thema's als gebied;
- moet er een (vaste) topografische ondergrond aanwezig zijn, zo ja welke? Standaard kan misschien een eenvoudige topografische kaart gebruikt worden, in grijstinten. Andere ondergronden, zoals luchtfoto's zijn optioneel (actie gebruiker);
- het systeem bekijkt of de bron restricties van thema's aangeeft. Als er bijvoorbeeld thema's aan elkaar gekoppeld zijn, moeten de gekoppelde thema's ook opgehaald worden en/of gevisualiseerd worden volgens de (gekoppelde) voorschriften.

3 Informatieanalyse thema's met behulp van meta-informatie

- het systeem verzamelt de voorschriften (met meerdere visualisatie-voorkeuren) die de bron geeft bij de thema's over kleur, geometrie, symboolvoorkeur, schaaldomein etc.;

- het systeem onderzoekt of er een relatie is tussen een ruimtelijk object en attributen, als er een attribuuttabel is moet het object ook aanklikbaar weergegeven worden zodat de objecten gerelateerd kunnen worden aan deze tabel;
- het systeem vergelijkt de schaaldomeinen van de thema's en bepaalt of alle thema's in de door de gebruiker gewenste schaal weergegeven kunnen worden;
- het systeem stelt een maximale en minimale schaal in waarin de huidige thema's nog afgebeeld kunnen worden (op basis van additionele informatie van de gegevensbron). Het overschrijden van die schaal zorgt voor actie van het systeem.

4 Keuze grafische variabelen

- Het systeem vergelijkt de voorschriften van de thema's;
- Zijn er twee of meer kleuren hetzelfde?;
- Zijn er twee of meer kleuren slecht onderscheidbaar?;
- Kunnen er thema's samengevoegd worden? (semantische generalisatie: wordt in dit afstudeeronderzoek niet nader onderzocht);
- Worden er in twee of meer thema's dezelfde symbolen gebruikt?;
- Als dit het geval is moet het systeem actie ondernemen;
- Zijn de voorschriften volgens de heersende conventies? (dit is zeer moeilijk automatisch te bepalen, hiervoor is een formele semantiek noodzakelijk);
- Zo nee, dan moet het systeem door middel van een waarschuwing aan de gebruiker aangeven dat de weergave afwijkt van de heersende conventies en dat de gebruiker op moet passen voor eventuele misleiding;
- Moet het systeem de voorschriften volgen of mag hij ze overrulen?
- Aangenomen wordt dat geen enkele bron de voorschriften verplicht op kan dringen, en dat de voorschriften altijd overtreden kunnen worden als dit in het belang is van een verantwoorde kartografische weergave of als de (expert) gebruiker dit aangeeft;
- Het systeem bepaalt of het kaartbeeld te vol is, als dat zo is moet het systeem actie ondernemen;
- Het systeem past bij in- of uitzoomen de regels van generalisatie toe voor zover de schaal dat toelaat. Kunnen deze regels niet meer toegepast worden moet het systeem lagen uit gaan zetten en de gebruiker informeren.

5 Legenda

- De gebruiker zet de af te beelden thema's op volgorde van belangrijkheid in de legenda, de vorm van weergave en de gebruikte kleur zijn ook terug te vinden in de legenda;
- De legenda kan door de gebruiker aangepast worden. De legenda vormt voor de gebruiker het centrale en belangrijkste interactie-element;
- Het verschuiven van de thema's in de legenda zorgt voor een verandering van de volgorde van belangrijkheid;
- Ook de originele bron moet in de legenda worden weergegeven zodat de kwaliteit en de betekenis van het thema achterhaald kan worden;
- Er moet een noordpijl of coördinatenraster aangezet kunnen worden, dit is voornamelijk belangrijk voor het plotten of printen van de visualisatie;
- er moet een schaalbalk weergegeven worden die verandert als de schaal verandert.

3.4.2 Uitwerking in conditionele regels

Deze regels worden nu uitgewerkt in IF …THEN constructies zoals deze in §3.1.1 besproken zijn. De hier uitgewerkte regels zijn de regels waarvan het beslisproces niet te complex is. Het symbool ➨ staat voor een handeling van het systeem: het doet iets of het stelt iets vast.

bepalen van de gewenste schaal

BENODIGDE INFORMATIE [gebied]
INFORMATIE-DOMEIN [coördinaten, bounding box, gebiedsnaam]

IF [gebied = coördinaten]
THEN [bepaal de schaal]
IF [gebied = bounding box]
THEN [bepaal de schaal waarbij de bounding box nog net in zijn geheel afgebeeld kan worden]
IF [gebied = gebiedsnaam]
THEN [bepaal de coördinaten]
AND [bepaal de gewenste schaal]
BENODIGDE INFORMATIE [gewenste schaal]
INFORMATIE-DOMEIN [schaal]
➨[schaal]

het kiezen en ophalen van de af te beelden thema's

➨[het systeem maakt een set van bronnen en thema's die gerelateerd zijn aan het opgegeven interessegebied, interactie gaat via de legenda]

BENODIGDE INFORMATIE [nieuw thema]
INFORMATIE-DOMEIN [keuze thema]
IF [thema op de n-de rij van de legenda]
THEN [set prioriteit = n]

➨ [legenda bepaalt de prioriteit, door slepen kan deze veranderen]

➨[browser haalt de gegevens op]

➨[lijst van thema's met prioriteiten]

het bepalen van de ondergrond

BENODIGDE INFORMATIE [welke ondergrond]
INFORMATIE-DOMEIN [luchtfoto, topografische ondergrond, anders]

IF [ondergrond = luchtfoto]
THEN [haal de luchtfoto op]
IF [ondergrond = topografisch]
THEN [haal de topografische kaart op]
IF [ondergrond = anders]
THEN [haal {anders} op]

het vaststellen van gegevensrelaties of visuele relaties tussen thema's

IF [thema x = gerelateerd aan thema y of thema's y, z, ..]
THEN [haal thema y of thema's y, z, .. ook op]
AND [informeer de gebruiker over de relatie tussen de thema's]
IF [thema x = geen relaties]
THEN [doorgaan]

het bepalen van de definitieve schaal

IF [schaal binnen schaalbereik thema x]
THEN [thema x kan afgebeeld worden]

IF [schaal buiten schaalbereik thema x]
THEN [ga naar oplossingsschema schaalconflict]

IF [geen bounding box]
THEN [schaal = schaal thema met kortste afstand gewenste schaal]
IF [bounding box]
THEN [schaal is gewenste schaal]

hoeveelheid data

➨[controleer wat de hoeveelheid data per oppervlakte-eenheid is]
IF [hoeveelheid data is te veel]
THEN [ga naar oplossingsschema te veel data per oppervlakte-eenheid]

➨[controleer of er binnen de andere thema's gegeneraliseerd moet worden aan de hand van de hoeveelheid data die afgebeeld dient te worden]

➨[voer de eventuele generalisatie uit]

➨[bepaal de schaalbereik waarbij alle afgebeelde thema's geschikt zijn]
schaalbereik = [grootste minimale schaal - kleinste maximale schaal]

➨[bij zoomen geldt dan]
IF [schaal buiten schaalbereik]
AND [thema generaliseerbaar]
THEN [thema generaliseren]
ELSE
IF [gelimiteerd]
THEN [thema weglaten met feedback]
ELSE [thema afbeelden met feedback]
IF [schaal binnen schaalbereik]
THEN [niets aan de hand]

weergave: kleuren

IF [kleur thema x = kleur thema y]
AND [geometrie thema x = geometrie thema y]
THEN [ga naar oplossingsschema kleurconflict]

IF [kleur thema x ≈ kleur thema y]
THEN [ga naar oplossingsschema kleurconflict]

weergave: symbolen

IF [symbool thema x = symbool thema y]
THEN [ga naar oplossingsschema symboolconflict]

generaliseren

➨[wordt niet uitgewerkt]

semantiek

➨[wordt niet uitgewerkt]

legenda

IF [gebruiker verandert de volgorde]
THEN [verander prioriteit: als het thema op de n-de rij in de legenda staat, dan prioriteit is n]
IF [volledig]
THEN [gebruiker mag weergave veranderen, mits met waarschuwingen van het systeem ➨ gebruik het oplossingsschema kleur- en symboolconflict]

IF [gelimiteerd]
THEN [gebruiker mag weergave niet aanpassen]

Oplossingsschema's

schaalconflict

➨voer generalisatie uit [niet uitgewerkt];
➨kijk of hetzelfde thema in een andere schaal opgehaald kan worden;
➨de gebruiker waarschuwen over het conflict. Weglaten of toch afbeelden?

te veel data

➨voer een generalisatie uit, zo mogelijk per probleemgebied;
➨kijk of het thema met een andere schaal opgehaald kan worden;
➨bereken de gemiddelde datahoeveelheid voor een kleiner gebied. Is de datahoeveelheid in meer dan 15% van de gebieden te veel dan mag het thema niet zomaar afgebeeld worden: lokaal aanpassen of globaal generaliseren;
➨als de hoeveelheid data lokaal te veel is beeld het thema af maar geef de gebruiker een waarschuwing;
➨als de hoeveelheid data globaal te veel is, de gebruiker waarschuwen over het conflict. Weglaten of toch afbeelden?

Het oplossingsschema voor conflicten worden pas in werking gesteld als de gebruiker alle gewenste thema's heeft opgevraagd. Dit om te voorkomen dat het kaartbeeld continu verandert door het toevoegen van een nieuw thema. Wat betreft kleur worden alle gekozen

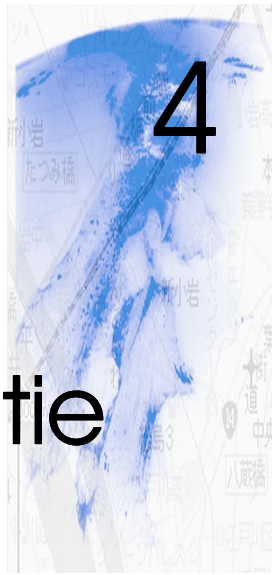
thema's dus in eerste instantie in de eerste voorkeursweergave afgebeeld. De schema's worden geactiveerd door middel van een knop [optimaliseren] in de legenda. Als deze knop wordt ingedrukt wordt de optimale kaart gemaakt voor de huidige combinaties van thema's.
Als een thema na bijvoorbeeld uitzoomen niet meer zichtbaar is, kan het gebeuren dat de huidige weergave niet (meer) optimaal is. De nieuwe optimalisatie treedt echter pas in werking op het moment dat de gebruiker opnieuw de knop indrukt.

kleurconflict

➡gebruik de presentatievoorschriften van de bron;
➡zet voor prioriteit = 1 weergavevoorkeur = eerste voorkeur [1.1].
➡color [1]
➡zet voor prioriteit = 2 weergavevoorkeur = eerste voorkeur [2.1].
IF [2.1] = color [1]
THEN [zet voor prioriteit = 2 weergavevoorkeur = tweede voorkeur [2.2].
➡color [2]
➡ zet voor prioriteit = 3 weergavevoorkeur = eerste voorkeur [3.1].
IF [3.1] = color [1] OR IF [3.1] = color [2]
THEN [zet voor prioriteit = 3 weergavevoorkeur = tweede voorkeur [3.2].
IF [3.2] =color [1] OR IF [3.2] = color [2]
THEN [zet voor prioriteit = 3 weergavevoorkeur = derde voorkeur [3.3].
➡color [3]
Als er thema's visueel gerelateerd zijn, is de weergave afhankelijk van het thema met de hoogste prioriteit. Op het moment dat dit thema een kleur toegewezen krijgt, krijgt het gerelateerde thema de kleur die daarbij hoort.
➡ ga door tot er geen conflicten meer zijn of..
➡ als er met de gegeven presentatievoorschriften geen oplossing gevonden kan worden, is het het beste een waarschuwing te geven over het voorkomen van een kleurconflict.

symboolconflict

➡ga terug naar de voorschriften van de bron;
➡kijk naar de semantische kenmerken van de conflicterende symbolen: zelfde semantiek dan samenvoegen in één thema/onderwerp met hetzelfde symbool;
➡kijk naar prioriteiten: hoog = voorkeur;
➡kijk naar conventies: past bij semantische kenmerken thema = voorkeur;
➡als [volledig] dan gebruiker vragen of thema aangepast moet worden;
➡als [gelimiteerd] en als [volledig zegt aanpassen]: ken een nieuw symbool toe aan thema, waarbij de nieuwe symbool een duidelijke referentie moet hebben naar het oude symbool;
➡stel de gebruiker op de hoogte van de aanpassing.



4 Implementatie

Voor de implementatie is gebruik gemaakt van bestaande software Magma/Lava, dat ontwikkeld is voor de gemeente Tilburg voor een datawarehouseproject. In dit afstudeeronderzoek zijn de gegevens ook beschikbaar gesteld door de gemeente Tilburg, aangevuld met gegevens van het Kadaster. Allereerst zal in § 4.1 het Tilburgse project toegelicht worden, daarna zal § 4.2 de software belichten alsmede de motivatie voor de keuze voor de gebruikte software en datasets. Daarna zal in § 4.3 de implementatie van één van de opgestelde regels voor kleurconflicten worden beschreven.

4.1 Het datawarehouse in Tilburg

De doelstelling van het datawarehouse-project van de gemeente Tilburg is het voorzien in de totale informatiebehoefte van de gemeente als geheel. Vanaf alle werkplekken is er toegang tot geïntegreerde vastgoedinformatie, luchtfoto's, geografie van wegen, panden en percelen foto's van monumentale gebouwen en tekstdocumenten waarin onder andere voorschriften voor bestemmingsplannen werden opgenomen. Bij overheden is er een grote vraag naar meervoudige informatie: niet alleen de locatie van een perceel moet opgevraagd kunnen worden, maar ook van wie het perceel is; bovendien moet achterhaald kunnen worden van wie alle naburige percelen zijn [Schouten, 2000].
Een datawarehouse bestaat uit een database waarin gekopieerde gegevens uit allerlei processystemen (of bronnen) opgeslagen zijn. De gegevens kunnen in samenhang bevraagd worden, ongeacht of de gegevens afkomstig zijn uit dezelfde bron of niet. Het gebruiken van het datawarehouse belast dus de originele bronnen niet (behalve bij het maken van een kopie), bovendien kunnen er makkelijk nieuwe objecten worden toegevoegd.

Het betreffende datawarehouse zal niet op internet maar op een lokaal netwerk (intranet) toegankelijk zijn. Toch geeft het een goede indruk van het werken met gegevens uit meerdere bronnen, zoals dat bij internet het geval zou zijn. De gegevens komen van verschillende afdelingen van de gemeente, iedere afdeling heeft zijn eigen manier van opslag en structurering. Bovendien werden in dit datawarehouse naast geografische bestanden ook luchtfoto's en thematische informatie opgenomen: gegevens die duidelijk een andere oorsprong en structuur kennen. Daarnaast was het bedrijf PGS (dat de software Magma/Lava voor het Tilburgse project ontwikkelde), bereid de software beschikbaar te

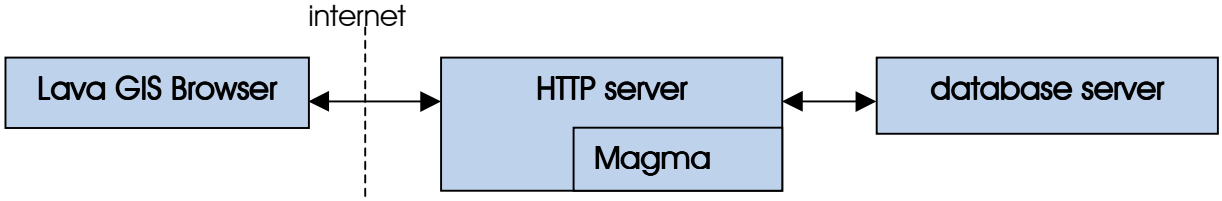
stellen, en waren de gemeente Tilburg en het Kadaster bereid de benodigde data beschikbaar te stellen.

4.2 Magma/Lava

Magma/Lava bestaat uit twee hoofdonderdelen:

- **Lava** het browsergedeelte, een Java applet dat geografische gegevens visualiseert als een kaart met meerdere kaartlagen;
- **Magma** een C++ server applicatie die webserver verzoeken doorstuurt naar de database server, die gegevens bevat die door Lava kunnen worden gevisualiseerd.

Figuur 4.1 geeft een overzicht van de Magma/Lava architectuur.



figuur 4.1: Magma/Lava architectuur

De **Lava-browser** is een kaart die bestaat uit verschillende lagen (LavaLayer), die één voor één getekend worden. Elke kaartlaag is een visuele representatie van een bepaald thema van de gegevens in de database. Hoe de verschillende lagen in Lava gevisualiseerd worden, wordt bepaald door een *LavaShape*, een Java klasse die de data van het type object omvat en methoden verzorgt om de laag in een kaart te tekenen. Er zijn meerdere typen LavaShapes, zoals *PolygonLavaShape* (die een polygoon representeert) en *PointLavaShape* (die een puntobject representeert). Nieuwe typen LavaShapes kunnen zelf worden toegevoegd om nieuwe objecttypen te definiëren.

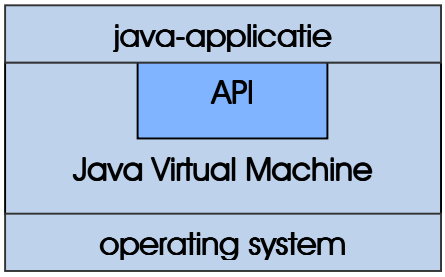
Een LavaShape specificeert alle kenmerken die belangrijk zijn voor het tekenen van het object:

- Constructie parameters, zoals de lijst van punten die nodig zijn om een polygoon weer te kunnen geven;
- Eigenschappen als kleur, vulpatroon, lijndikte etc. Deze eigenschappen kunnen door de gebruiker veranderd worden. De hier vast te stellen eigenschappen worden gebruikt voor de hele laag, dus elk LavaShape object in deze laag gebruikt deze eigenschappen;
- Attributen, die door de server worden geleverd. Dit kunnen bijvoorbeeld attribuuttabellen zijn, foto's van objecten in de kaart, en links naar bijvoorbeeld een bestemmingsplan voor een bepaald gebied in de vorm van een tekstdocument;
- Wat er moet gebeuren als het object wordt aangeklikt. Dat kan het opkomen van een dialoogbox zijn, het weergeven van informatie over het object of het openen van een webpagina met gerelateerde informatie.

De gegevens die nodig zijn om een bepaalde laag te visualiseren worden via het internet opgehaald bij verschillende Magma-servers. Een Magma server vertaalt een vraag binnengekomen via de webserver van een Lava client (bijvoorbeeld als er een nieuwe laag wordt aangezet) naar een SQL query op een onderliggende database. Daarnaast gebruikt

Magma meta-data die informatie verstrekt over beschikbare Magmaservers, welke informatie beschikbaar is en configuratie-opties voor Lava, bijvoorbeeld welke kleuren er gebruikt worden en voor welke schaal de gegevens geschikt zijn. Deze meta-data zelf wordt opgeslagen is in de configuratiefile, die gelezen wordt op het moment dat de Lava-browser opstart. Binnen de Lava broncode is aangegeven welke onderdelen er gespecificeerd kunnen worden in de configuratiefile.

Lava kan gerund worden als Java applicatie of als Java applet. Een Java omgeving bestaat uit de Java Application Programming Interfaces (APIs) en de Java Virtual Machine (JVM) (zie figuur 4.2).



figuur 4.2 Java, API en Java Virtual Machine

Java APIs zijn interfaces van de bibliotheken van gecompileerde code die gebruikt kunnen worden in Java-programma's. Deze standaard functionaliteiten hoeven dus niet iedere keer apart geprogrammeerd te worden, bovendien zijn ze uitbreidbaar. Java applicaties worden geinterpreteerd door de Java VM van het operating systeem. Dit betekent dat ieder computersysteem waarop de Java VM geïnstalleerd is Java applicaties kan starten, ongeacht het originele systeem waarop de applicatie is ontwikkeld. Met andere woorden: Java is platform-onafhankelijk, mits op het systeem waar de Java-applicatie gestart wordt de bijbehorende Java VM is geïnstalleerd. Er zijn twee versies van Java en de bijbehorende VM beschikbaar: Java 1 (ook wel Java 1.1) en Java 2 (voorheen Java 1.2, de laatste versie is 1.3 (september 2000)). Het grootste verschil tussen Java 1 en Java 2 is dat er voor Java 2 een plug-in geïnstalleerd moet worden, terwijl Java 1 overal aanwezig kan zijn, en bovendien zeer snel en betrouwbaar is. Java 2 biedt daarentegen meer mogelijkheden voor 2D grafische toepassingen.
Java is dus bijzonder geschikt voor het gebruik op internet, aangezien er geen rekening gehouden hoeft te worden met de vele operating systemen die de gebruikers van de applicatie zouden kunnen hebben. Het uiterlijk van de applicatie en de functionaliteit is echter wel afhankelijk van de versie van de browser en de VM.

4.3 Implementatie

In dit afstudeeronderzoek is een begin gemaakt van de ontwikkeling van een Kartografisch Expertsysteem voor een internet GIS met gedistribueerde bronnen. Gezien de complexiteit van het onderwerp wordt er bij de implementatie niet gestreefd naar volledigheid, maar naar het duidelijk maken van de werking van een KES zoals dat in de toekomst ontwikkeld zal moeten worden. In § 4.3.1 wordt beschreven welke regel er geïmplementeerd is. Het toevoegen van meerdere voorkeursvisualisaties wordt behandeld in § 4.3.2, waarna in § 4.3.3 wordt beschreven hoe de prioriteiten aan de verschillende lagen worden toegekend. Op basis van de prioriteiten worden de eigenschappen toegewezen, dit wordt beschreven in § 4.3.4. Tenslotte wordt in § 4.3.5 een voorbeeld gegeven van een optimalisatie van verschillende lagen.

4.3.1 Implementatie in de Lava-broncode

Voor de implementatie is ervoor gekozen om regels toe te voegen die ervoor zorgen dat er geen conflicten ontstaan op het gebied van kleurgebruik. Dit omdat de conflicten die hierbij op kunnen treden heel duidelijk zijn, het in heel duidelijke regels te omvatten is en omdat de verbetering ook direct zichtbaar is.
Zoals in § 3.4 al omschreven, wordt aangenomen dat de verschillende thema's visualisatievoorkeuren meesturen in de vorm van meta-data. De configuratiefile (genoemd in § 4.2) speelt hierbij een centrale rol (letterlijk en figuurlijk) als meta-databestand, waarin de verschillende voorkeuren worden vastgelegd. Bij het weergeven van meerdere gekozen thema's tegelijk kan het voorkomen dat twee thema's dezelfde eerste voorkeurskleur hebben. Op basis van prioriteit die de gebruiker aangeeft door lagen in de legenda in een bepaalde volgorde te plaatsen, bepalen de regels welke thema's aangepast moeten worden door vergelijking met de kleuren van andere thema's. Om te vermijden dat iedere keer na het toevoegen van een nieuwe laag de kleuren vanzelf veranderen, is ervoor gekozen om de regels pas in werking te stellen als de gebruiker dat aangeeft.
Bij het implementeren zijn de volgende uitgangspunten gehanteerd:

- De presentatie-voorschriften (meta-data) komen uit de configuratiefile die gebruikt wordt om de huidige Tilburg data weer te geven;
- De regels worden hard gecodeerd in de Java-code (dus niet als zelfstandig werkend expertsysteem) en gebruiken de gegevens uit de configuratiefile;
- De regels worden pas geactiveerd als de gebruiker op een knop drukt: Optimaliseer de presentatie.

4.3.2 Toevoegen van meerdere voorkeuren voor visualisatie

De regels die van belang zijn om tot een visualisatie zonder kleurconflict te komen gebruiken de meta-data die de bron meestuurt. Voor kleur zijn dan de visualisatievoorkeuren van belang. In dit geval wordt de meta-data geleverd door de configuratiefile. In de configuratiefile [zie ook appendix B] zien we het volgende:

```
Layer(
    name = "HuisnummerAdressen",
    relation = lookup("www.gdmc.nl/cgi-bin","tilburg","HUISNUMMERSMGM"),
    lavaShapeSpec=(
        type = "PointLavaShape"
        arguments = ("HUISNUMMERSATTR")
        properties = PropertyList("color=0,Shape=4"),
        attributes=
        AtttributeList("'prefix:','http://www.gdmc.nl:81/info/vraagadres?adres
        =','links:',ADRESNR,'target:','main'")
    ),
    visible = false,
    scaleRange=(from=0.01,to=2.5)
    colorMap = lookup("standard")
    )
```

De eigenschappen van deze kaartlaag worden aangegeven door de *properties*, als variabele van de Java-klasse *PropertyList*. Omdat dit een type *PointLavaShape* is kent deze laag alleen de eigenschappen *color* en *shape* (symbooltype weergegeven door een nummer). Voor het type PolygonLavaShape is deze lijst uitgebreider: eigenschappen als *fillpoly*, *borderwidth*, *bordercolor* en *pickable* kunnen worden toegevoegd. In dit geval is alleen de eigenschap *color* (en voor polygonen *bordercolor*) van belang.

Er zijn twee opties om een tweede en derde lijst van nieuwe eigenschappen (tweede en derde voorkeur) te definiëren:

1. Omdat PropertyList een klasse is van eigenschappen van de lagen, kunnen er nieuwe variabelen gedeclareerd worden (propertiesTwo, propertiesThree) die ook onderdeel zijn van die klasse;
2. In de lijst met eigenschappen wordt meteen de tweede en derde voorkeurskleur gedefinieerd: "colorTwo=3, colorThree=5".

Voor de huidige implementatie is de eerste optie gebruikt omdat deze een betere scheiding aangeeft tussen de verschillende voorkeurseigenschappen. Bovendien is er een koppeling van de gerelateerde eigenschappen: een tweede voorkeur kan In de configuratiefile worden de extra voorkeurseigenschappen als volgt beschreven.

```
Layer(
    name = "HuisnummerAdressen",
    relation = lookup("www.gdmc.nl/cgi-bin","tilburg","HUISNUMMERSMGM"),
    lavaShapeSpec=(
           type = "PointLavaShape"
           arguments = ("HUISNUMMERSATTR")
           properties = PropertyList("color=0,Shape=4"),
           propertiesTwo = PropertyList("color=3,Shape=4"),
           propertiesThree = PropertyList("color=2,Shape=4"),
          attributes = AttributeList("'prefix:',
          'http://www.gdmc.nl:81/info/vraagadres?adres=','links:',ADRESNR,
          'target:','main'")
        ),
    visible = false,
    scaleRange=(from=0.01,to=2.5)
    colorMap = lookup("standard")
    )
```

Binnen de Lava-broncode moeten voor propertiesTwo en propertiesThree dezelfde voorwaarden gelden die voor properties ook gelden. Voor alle methoden die een verwijzing naar *properties* hebben moeten dus methoden worden opgesteld die dezelfde verwijzingen hebben naar respectievelijk *propertiesTwo* en *propertiesThree*. De veranderingen in de betreffende java files zijn te vinden in appendix C.

4.3.3 Bepalen van de prioriteiten van de verschillende lagen

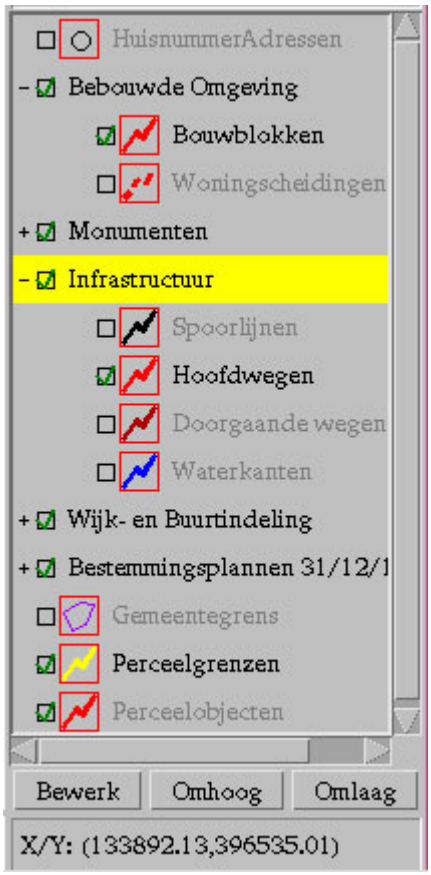
Bij het opstarten van het applet krijgen alle lagen in eerste instantie de eigenschappen *properties* toegewezen. De tweede en derde voorkeur *propertiesTwo* en *propertiesThree* worden wel als meta-data ingelezen maar nog niet geactiveerd. De gebruiker moet nu eerst zijn prioriteiten toewijzen zodat bepaald kan worden welke lagen het belangrijkst zijn. Dit gebeurt door de volgorde van de lagen in de legenda te veranderen, waarbij de laag die bovenaan de legenda staat de hoogste prioriteit heeft. Een laag kan verplaatst worden door deze te selecteren in de legenda en dan op de knop omhoog of omlaag te drukken.
Door het verplaatsen van lagen in de legenda krijgt deze laag een ander nummer in de index van de legenda. De bovenste laag krijgt indexnummer 0, de daaropvolgende laag heeft indexnummer 1 enzovoorts. Hoe hoger de laag in de legenda staat, hoe lager het indexnummer en hoe hoger de prioriteit. De lagen kunnen tussendoor steeds verplaatst worden door de gebruiker, zodat de prioriteit op een interactieve manier veranderd kan

worden. De methode die voor het toekennen van de prioriteiten is geïmplementeerd heet *setPriorities* (zie volgend kader), de volledige java-files zijn te bekijken in appendix C.

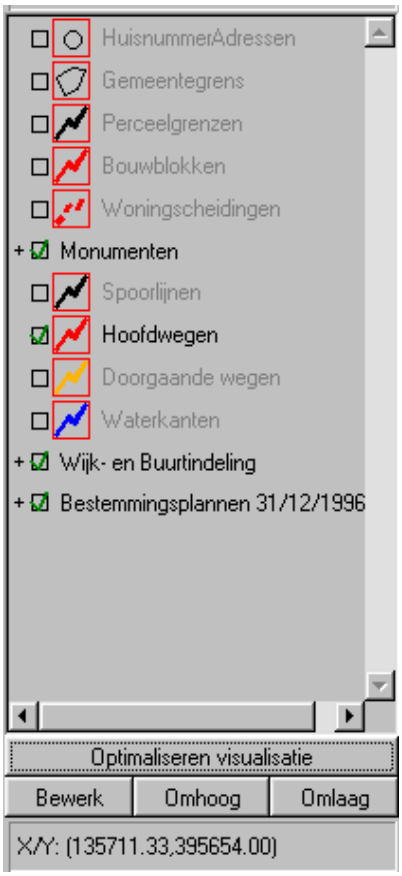
```
//-----------------setPriorities-------------
    public boolean setPriorities(Layer l)
    {
      Vector layerVec = findLayerVector(l);
      if (layerVec == null)
        return false;

     int index = layerVec.indexOf(l);
       l.prior = index + 1;
    return true
    }
```

4.3.4 Toewijzing van eigenschappen aan de hand van de gestelde prioriteiten

Op basis van de prioriteiten die de gebruiker stelt moet de methode *optimalProperties* de onderlinge eigenschappen (in dit geval de waarde van *color*) vergelijken om te zien of ze mogelijk een conflict op kunnen leveren. De methode *optimalProperties* wordt in werking gesteld op het moment dat de gebruiker de optimaliseerknop in de legenda indrukt. In figuur 4.3 is zowel de oude legenda als de nieuwe legenda (inclusief optimaliseerknop) te zien.



figuur 4.3a De oude legenda
Bron: Lava GIS Browser



figuur 4.3b De nieuwe legenda
Bron: Lava GIS Browser, development versie

De optimaliseerknop werkt vooralsnog alleen per geselecteerde laag. De methode *optimalProperties* roept als eerste de methode *setPriorities* aan om de huidige prioriteiten van de lagen te bepalen. Bij het toekennen van de eigenschappen begint de methode met de laag met de hoogste prioriteit (prioriteit één, deze krijgt altijd de eerste voorkeur). Daarna wordt aan de tweede laag (met prioriteit twee) ook de eerste voorkeur toegekend. De waarden van *color* worden onderling vergeleken: als ze niet dezelfde waarde hebben kan de laag met tweede prioriteit de eerste voorkeur behouden, hebben ze dezelfde waarde dan wordt aan de tweede laag de tweede voorkeursweergave toegewezen: *propertiesTwo*. Voor de derde laag wordt eveneens de eerste voorkeur toegewezen. De waarde van *color* wordt nu vergeleken met de waarden van *color* van de voorgaande lagen. Als de kleuren overeenkomen wordt de tweede voorkeursweergave toegewezen, daarna wordt de waarde van *color* van de derde laag weer vergeleken. Mocht deze waarde overeenkomen met een waarde van één van de voorgaande lagen wordt de derde voorkeursweergave toegewezen. Voor de opeenvolgende lagen wordt het proces herhaald. Door het toekennen van de nieuwe eigenschappen moeten de lagen opnieuw getekend worden met de veranderde kleuren.
Na het toekennen van een andere set van eigenschappen aan een laag(bijvoorbeeld *propertiesTwo*) heten de huidige eigenschappen van deze laag binnen het programma *properties*. Deze heeft dan de waarden zoals ze in *propertiesTwo* beschreven staan. Als de prioriteit van de laag verandert kan het gebeuren dat de eerste voorkeur weer mogelijk is, maar deze zijn niet meer terug te vinden omdat de huidige *properties* nog de waarden van propertiesTwo bevat. Daarom is, naast *propertiesTwo* en *propertiesThree*, ook *propertiesOne* toegevoegd dat dezelfde waarden heeft als de initiële waarden van *properties*. De waarden van properties kunnen dus veranderen, de waarden van *propertiesOne*, *propertiesTwo* en *propertiesThree* kunnen na het inlezen niet meer veranderen.
Na iedere verandering in de laagvolgorde in de legenda moet de optimaliseerknop opnieuw ingedrukt worden omdat de prioriteiten veranderd zijn. Het hele proces wordt dan opnieuw doorlopen. De methode *optimalProperties* ziet er als volgt uit:

```
//---------optimalProperties--------------------

  public boolean optimalProperties(Layer layer) {
       setPriorities (layer);

     if (layer.prior == 1){
           layer.properties = layer.propertiesOne;
           colorOne = layer.properties.value[0];
                }
     if (layer.prior == 2){
            {
                layer.properties = layer.propertiesOne;
                colorTwo = layer.properties.value[0];
            }
            if (colorTwo == colorOne) {
            layer.properties = layer.propertiesTwo;
            int colorTwo = layer.properties.value[0];
               }
        }
     if (layer.prior == 3){
            {
                layer.properties = layer.propertiesOne;
                colorThree = layer.properties.value[0];
            }
            if ((colorThree == colorOne) || (colorThree == colorTwo)) {
            layer.properties = layer.propertiesTwo;
            colorThree = layer.properties.value[0];
                }
            if ((colorThree == colorOne) || (colorThree == colorTwo)) {
            layer.properties = layer.propertiesThree;
            colorThree = layer.properties.value[0];
            }
        }
```

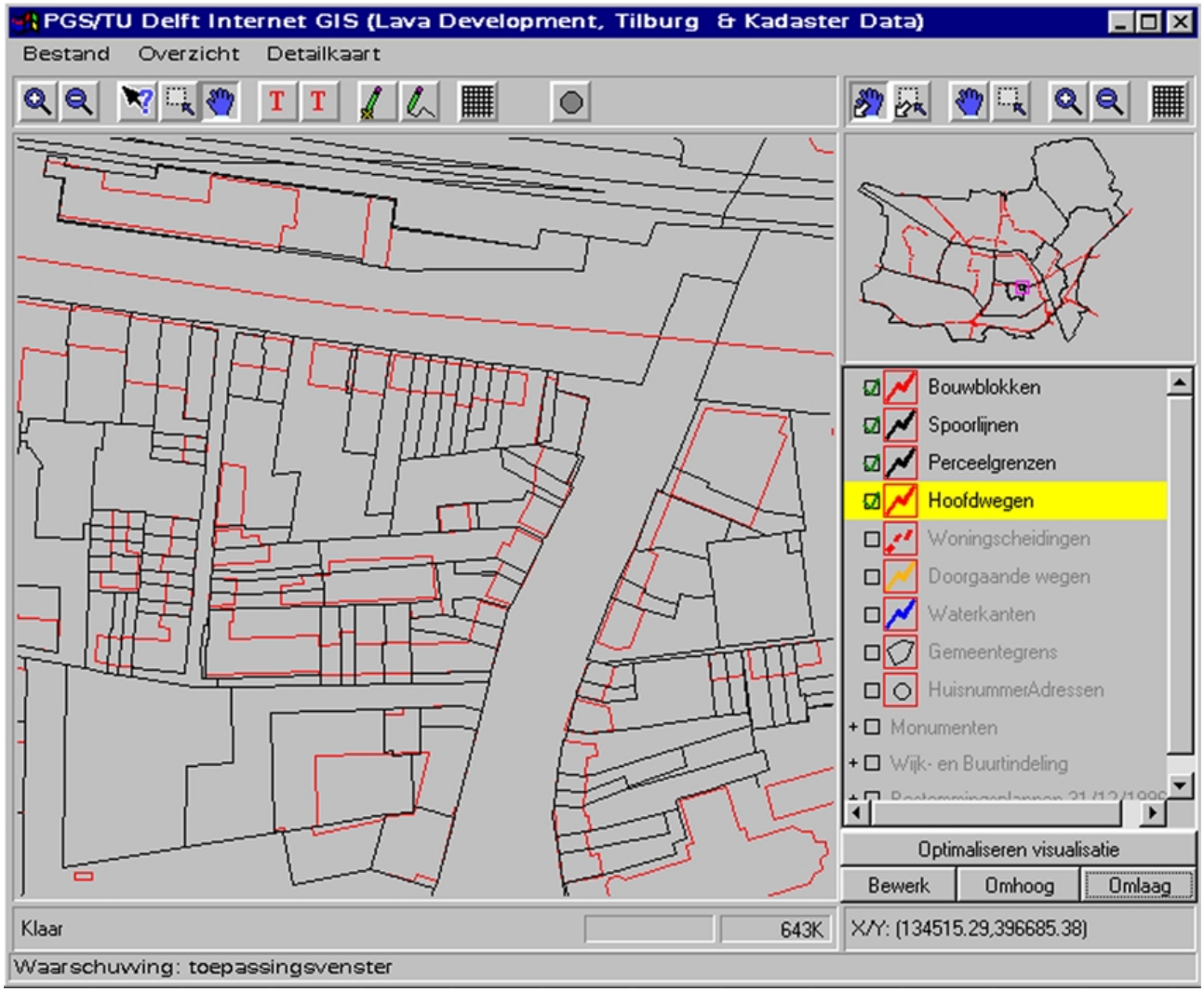
De volledige java-file is te vinden in appendix C.

4.3.5 Voorbeeld van een optimalisatie

De aangebrachte veranderingen tonen aan dat het systeem om kan gaan met meerdere voorkeursvisualisaties, deze onderling kan vergelijken en een optimale visualisatie kan voorschrijven. Er zijn echter enkele beperkingen die niet binnen het tijdsbestek beschikbaar voor implementatie konden worden opgelost. Deze beperkingen zijn:

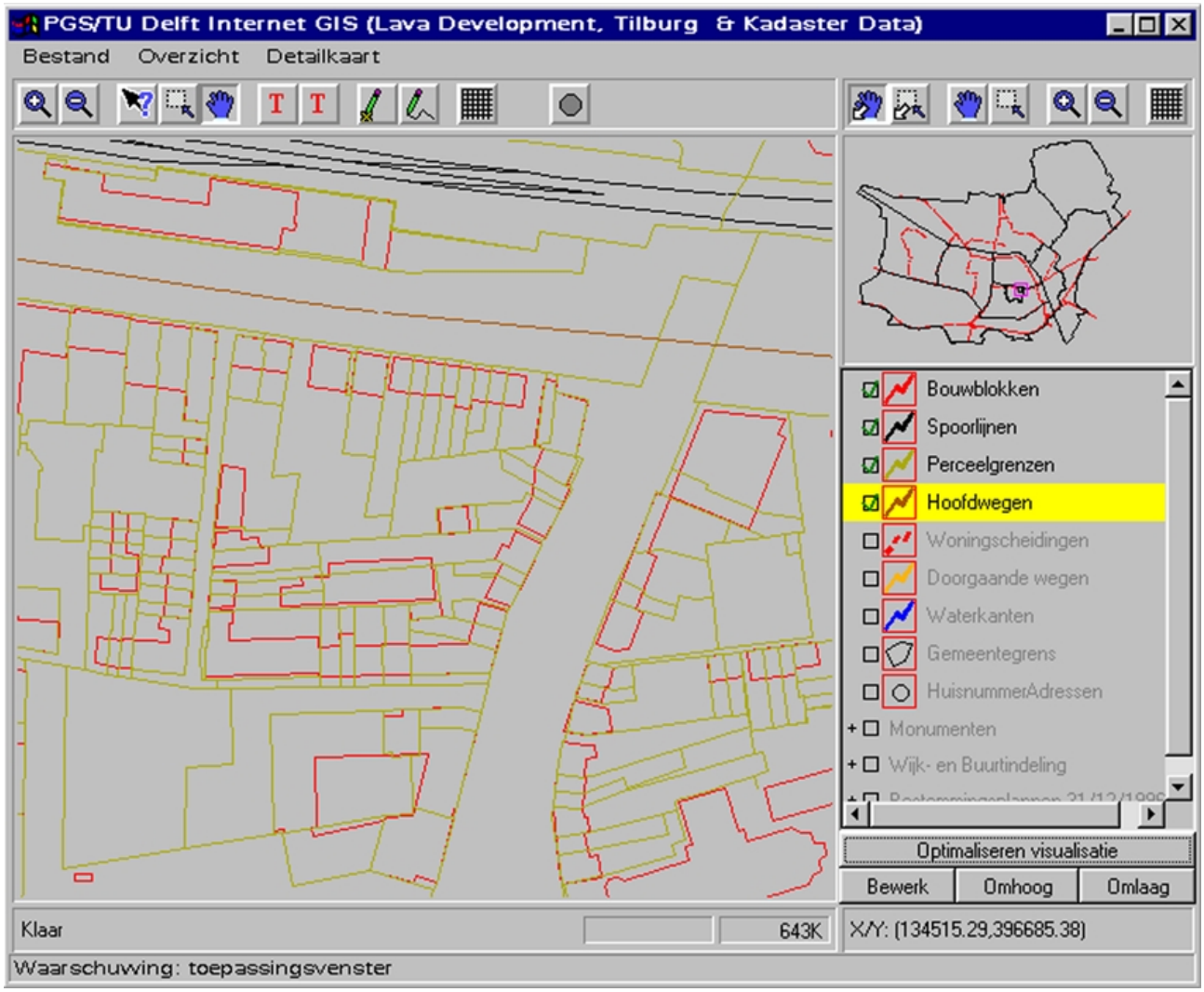
- De optimaliseerknop werkt alleen per laag, iedere laag moet dus apart geselecteerd worden en geoptimaliseerd worden;
- De indexering van de legenda (van belang voor het toewijzen van prioriteiten) is niet opvolgend als de lagen genest zijn: de hoofdlaag in de legenda is een lege laag, de geneste lagen hebben een eigen indexering die weer bij het begin begint. Om de werking van *optimalProperties* te demonstreren is de nesting in de legenda verwijderd. De nesting is echter belangrijk om thema's aan elkaar te kunnen relateren (bijvoorbeeld de relatie bouwblokken en woningscheidingen in het hoofdthema 'bebouwde omgeving'). De lagen die onder deze hoofdlaag liggen moeten dan beiden dezelfde prioriteit toegekend worden, zodat ze ook visueel aan elkaar gerelateerd zijn. Bij het verder ontwikkelen van deze methode is het belangrijk dat de nesting behouden blijft, terwijl er een juiste indexering plaatsvindt;
- De methode optimalProperties werkt vooralsnog alleen goed bij lijntypen (*PolylineLavaShapes*). De toekenning van nieuwe eigenschappen aan punttypen en vlaktypen (respectievelijk *PointLavaShapes* en *PolygonLavaShapes*) gaat in een aantal gevallen niet goed. Binnen dit tijdsbestek was echter niet te achterhalen waarom de toewijzing van een nieuwe set van eigenschappen bij deze typen niet correct verloopt. De verwachting is dat de aanpassingen die dit probleem verhelpen wel gemaakt kunnen worden;
- Er worden drie verschillende sets van eigenschappen gedefinieerd voor iedere laag. Er wordt vanuit gegaan dat dit voldoende is om aan iedere laag een unieke kleur toe te kennen. Het kan natuurlijk gebeuren dat er een laag is die na toewijzing van alle mogelijke voorkeuren nog steeds een kleurconflict heeft met een laag die een hogere prioriteit heeft. In dit geval krijgt de betreffende laag de derde prioriteit als definitieve toegekend. Bij verdere ontwikkeling moet hiervoor een oplossing gevonden worden: bijvoorbeeld door de gebruiker te waarschuwen, of door een maximaal aantal zichtbare lagen voor te schrijven. Het uitbreiden van het aantal sets van eigenschappen (*propertiesFour*, *propertiesFive* etc.) heeft niet de voorkeur.

Hierna volgt een voorbeeld van een mogelijke optimalisering van het kaartbeeld.



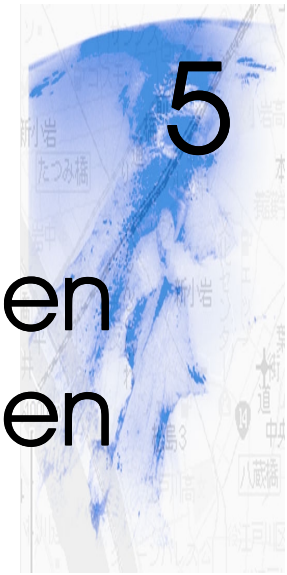
figuur 4.4 Lava screendump vóór optimalisatie

Er zijn 4 lagen geselecteerd die worden weergegeven. Daarvan hebben 2 lagen de kleur zwart (percelen en spoorlijnen) en 2 lagen de kleur rood (bouwblokken en hoofdwegen). Er is een kleurconflict: in de kaart is niet meer te onderscheiden welke grafische elementen nu wat uitbeelden. Door de volgorde in de legenda geeft de gebruiker aan welke lagen het belangrijkst zijn. Op grond hiervan wordt de kaart geoptimaliseerd. Na het selecteren van de lagen en het drukken van de optimaliseerknop verandert de kaart (zie de volgende pagina).



figuur 4.5 Lava screendump na optimalisatie

Omdat de lagen bouwblokken en spoorlijnen hoger in de prioriteitenlijst staan, hebben deze hun eigen kleur behouden. De lagen percelen en hoofdwegen hebben hun kleur aangepast aan de tweede voorkeur. Hierdoor is het onderscheid in de kaart goed te maken, en is de kaart kartografisch verbeterd.



5 Conclusies en aanbevelingen

Het doel van dit afstudeeronderzoek is een begin te maken van een Kartografisch Expertsysteem voor het gebruik met een multi-bron internet GIS. Het gebruik van gedistribueerde bronnen biedt de mogelijkheid om meerdere datasets tegelijk te bevragen en te visualiseren, terwijl de GIS functionaliteit zich bij de gebruiker bevindt. Bij de visualisatie van de gegevens kunnen echter gemakkelijk conflicten ontstaan, omdat naast de data zelf geen informatie wordt verstrekt over hoe deze weergegeven dient te worden. Naast het feit dat de bronnen die op het internet hun gegevens verstrekken ook de visualisatievoorschriften moeten leveren, is het noodzakelijk een set van regels te ontwikkelen die de gebruiker kunnen begeleiden en adviseren. De gebruiker kan hiermee geholpen worden om te komen tot een juiste kartografische visualisatie. In dit afstudeeronderzoek is onderzocht wat de huidige stand van zaken is op het gebied van multi-bron internet GIS, en wat de mogelijkheden zijn voor het ontwikkelen van een kartografisch expertsysteem dat de gebruiker begeleidt bij het visualiseren van de gegevens uit gedistribueerde bronnen. Daarna is een gedeelte van de ontworpen regels geïmplementeerd in een bestaand internet-GIS programma (Magma/Lava), geschreven in Java.

5.1 Conclusies

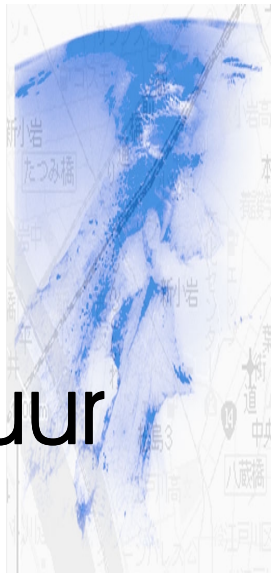
Uit het onderzoek zijn de volgende conclusies getrokken:

- Huidige ontwikkelingen op het gebied van internet en de interoperabiliteit van gegevensuitwisseling vragen om een gestandaardiseerd internet GIS voor gedistribueerde bronnen. Uit de literatuur blijkt dat vele instellingen, zowel bedrijven als overheid, interesse hebben in een dergelijk systeem.
- Het OpenGIS Consortium legt hiervoor een goede basis met de opzet van het WebMapping Testbed en andere standaards voor interoperabiliteit. De eerste versie van deze standaards is gereedgekomen in het voorjaar van 2000, en de verschillende participanten van het OpenGIS Consortium zullen de ontwikkeling voortzetten over hoe deze standaard zijn uiteindelijke vorm krijgt.
- De architectuur die het meest geschikt is voor het gebruik van gedistribueerde bronnen wordt door het Open GIS Consortium omschreven als 'data-case': de server stuurt

slechts de gegevens zelf (of het DLM inclusief visualisatievoorschriften) op, zodat de gebruiker geheel vrij is om zijn eigen visualisatie (of DKM) te maken.
- In dit kader is ook het onderscheid in DLM en DKM van belang: waar voorheen de gegevens en grafische presentatie samengingen, is bij gegevensuitwisseling via de 'data-case' een duidelijke scheiding gekomen tussen de gegevens zelf en de grafische presentatie.
- Uit dit onderzoek (maar ook uit eerdere onderzoeken) blijkt dat het ontwikkelen van generieke regels voor sommige toepassingen, zoals voor generalisatie en semantische aspecten, zeer complex is. Voor eenvoudiger toepassingen, zoals het aanpassen van kleuren en het bepalen van weergaveschaal is het mogelijk om regels te ontwerpen zoals die in een KES gebruikt kunnen worden.
- In een internet GIS speelt de legenda een belangrijke rol. De gebruiker kan hierin aangeven wat zijn prioriteiten zijn (en zo impliciet zijn doelen aangeven), bovendien kan de legenda de gebruiker informeren over de huidige visualisatie door middel van kleur, iconen, highlights etc.
- De gebruikte programmeertaal Java door zijn platformonafhankelijkheid zeer geschikt voor het gebruik voor internet GIS.

5.2 Aanbevelingen

- Het gebruik van internet GIS zal alleen maar toenemen. Verder onderzoek is noodzakelijk om de huidige ontwikkelingen te volgen en mogelijke nieuwe toepassingen te ontwikkelen.
- De eerste versie van de standaards van het Open GIS Consortium zijn in het voorjaar van 2000 gereedgekomen. De verschillende participanten van het OGC zullen de ontwikkeling voortzetten over over hoe deze standaard zijn uiteindelijke vorm krijgt.
- Voor een goede ontwikkeling van interoperabiliteit moeten ook de leveranciers van geografische informatie zich aanpassen aan de huidige ontwikkelingen en de gegevens leveren zoals de standaards van het OGC beschrijven. Ook overzichtelijkheid van de eigen gegevens is hierbij van belang.
- Nieuwe ontwikkelingen op het gebied van XML, GML en XSLT kunnen mogelijk een rol spelen bij het uitwisselen van gegevens. Verwacht wordt dat deze protocollen de nieuwe standaard gaan worden voor het versturen van query's (HTTP - GetMap), het DLM (GML) en de visualisatievoorschriften (XSLT).
- In dit onderzoek is niet verder ingegaan op de meest complexe onderdelen van de beeldschermkartografie: semantiek, generalisatie en labelplaatsing. Deze onderwerpen moeten verder onderzocht worden zodat hiervoor ook regels kunnen worden opgesteld die later aan het KES kunnen worden toegevoegd.
- Het KES is hier verder niet ontwikkeld als apart regelgebaseerd expertsysteem, maar hard gecodeerd in de broncode. Verder onderzoek naar het gebruik van expert systemen is noodzakelijk om het KES uiteindelijk als apart onderdeel te kunnen implementeren in het internet GIS.



Literatuur

Adler, S., A. Berlund et.al., **Extensible Stylesheet Language (XSL) version 1.0** . W3C working draft, 18 -10-2000.
http://www.w3.org/TR/xsl

Andrienko, G., N. Andrienko, H. Voss, J. Carter, **Internet mapping for dissemination of statistical information**. Computers, Environment and Urban Systems, no. 23 (1997), pp. 425-441.

Andrienko, G., N. Andrienko, **Visual data exploration by dynamic manipulation of maps.** 8th International Symposium on Spatial Data Handling, Vancouver, July 11-15 (1998), pp. 533-542.

Andrienko, G., N. Andrienko, **Interactive maps for visual data exploration.** International Journal of Geographical Information Science, vol. 13, no. 4. (1999), pp. 355-374.

Andrienko, G. en N. Andrienko, **Making a GIS intelligent: CommonGIS Project view.** AGILE conference, Rome 1999.

Arleth, M., **Problems in screen map design.** 19th International Cartographic Conference with ICA, Ottawa, August 14-21 (1999), session 01-D.

Bertin, J., **Semiology of graphics (vertaald door William J. Berg).** Londen, The University of Wisconsin Press, 1983.

Bishr, Yaser A., Hardy Pundt, Christoph Ruther, **Proceeding the road of semantic interoperability: Design of a semantic mapper based on a case-study from transportation**. In: A. Vçkovski (editor), Interoperable and distributed processing in GIS. Londen: Taylor & Francis, 1998, pp. 203-216.

Boye, Janus, **XML, what's in it for us.** 1998.
http://www.tech.irt.org/articles/js072/index.htm

Bray, Tim, Jean Paoli, C.M. Sperberg-McQueen, **Extensible Markup Language (XML) 1.0 (Second Edition)**. W3C recommendation, 6-10-2000.

http://www.w3.org/TR/2000/REC-xml-20001006

Brown, Carol E., Daniel E. O'Leary, **Introduction to artificial intelligence and expert systems.** 2000.
http://www.bus.orst.edu/faculty/brownc/es_tutor/es_tutor.html

Buttenfield, Barbara P., David M.. Mark, **Expert systems in cartographic design.** In: D.R. Fraser (editor), Geographic Information Systems: the microcomputer and modern cartography. Oxford: Pergamon Press, 1991, pp. 129-150.

Buttenfield, Barbara P., David M.. Mark, **Sharing vector geospatial data on the internet.** 19th International Cartographic Conference with ICA, Ottawa, August 14-21 (1999), session 10-C.

Campione, Mary, Kathy Walrath**, The Java tutorial Second Edition: Object-oriented programming for the internet.** California: Addison Wesley Longman, 1998.

Cartwright, W., **Extending the map metaphor using web delivered multimedia.** International Journal of Geographical Information Science, vol. 13, no. 4. (1999), pp. 335-353.

Cecconi, A., C. Shenton, R. Weibel, **Tools for cartographic visualisation of statistical data on the internet.** 19th International Cartographic Conference with ICA, Ottawa, August 14-21 (1999), session 07-C.

Clarke, Keith C., **Analytical and computer cartography (2nd edition).** Englewood Cliffs : Prentice-Hall, 1994.

Doyle, A., **OpenGIS Web Map server interface implementation specification (revision 1.0.0).** OpenGIS project document 00-028, 2000.
http://www.opengis.org/techno/specs/00-028.pdf

Elzakker, C. Van, **Thinking aloud about exploratory cartography.** 19th International Cartographic Conference with ICA, Ottawa, August 14-21 (1999), session 34-A.

Fraser Taylor, D.R., **Geographic Information Systems: the microcomputer and modern cartography.** In: D.R. Fraser (editor), Geographic Information Systems: the microcomputer and modern cartography. Oxford: Pergamon Press, 1991, pp. 1-20.

Gehrels, B., **Combining maps on the web.** Geo-informatics, jan/feb (2000), pp. 12-15.

Grelot, Jean-Philippe, **Cartographers and microcomputers.** In: D.R. Fraser (editor), Geographic Information Systems: the microcomputer and modern cartography. Oxford: Pergamon Press, 1991, pp. 237-247.

Herbert, Jeff, Chuck Murray, **Oracle Spatial User's Guide and Reference**. Release 8.1.6, Part. no A77132-01, 1999.

Jones, Christopher, **Geographical Information Systems and Computer Cartography.** Singapore: Addison Wesley Longman, 1997.

Kähkönen, J., L. Letho, T. Kilpeläinen, T. Sarjakoski, **Interactive visualisation of geographical objects on the internet.** International Journal of Geographical Information Science, vol. 13, no. 4. (1999), pp. 429-438.

Keats, J.S., **Understanding Maps (2nd edition).** Harlow: Longman Higher Education, 1996.

Keller, Stefan F., Hugo Thallmann, **Modeling and sharing graphic presentations of geospatial data.** In: A. Vçkovski (editor), Interoperable and distributed processing in GIS. Londen: Taylor & Francis, 1998, pp. 151-162.

Kingston, R., S. Carver, A. Evans, I. Turton, **Web based public participation geographical information systems.** Computers, Environment and Urban Systems, no. 24 (2000), pp. 109-125.

Kraak, M., A. MacEachren, **Visualisation for exploring of spatial data.** International Journal of Geographical Information Science, vol. 13, no. 4. (1999), pp. 285-287.

Kraak, M., R. Hootsmans**, National Mapping Organisations and the WWW, challenges and opportunities.** 19th International Cartographic Conference with ICA, Ottawa, August 14-21 (1999), session 03-D.

Lake, Ron, Adrian Cuthbert (editors), **Geography Markup Language.** version 1.0, 2000.
http://www.opengis.org/techno/specs/00-029/GML.html

Lehto, L., **XML in Web-base geospatial applications.** 3rd AGILE conference on Geographic Information Science, Helsinki-Espoo, May 25-27 (2000), pp. 162-167.

MacEachren, Alan M., **Some truth with maps: a primer on symbolisation and design.** Washington : Association of American Geographers, 1995.

Morgenstern, D., D. Schurer, **A concept for model generalization of digital landscape models from finer to coarser resolution.** 19th International Cartographic Conference with ICA, Ottawa, August 14-21 (1999), session 31-D.

Ramirez, J. Raul, **Maps for the future: a discussion.** 19th International Cartographic Conference with ICA, Ottawa, August 14-21 (1999), session 22-A.

Rengelink, Maureen, **Ontwikkeling van een Cartografisch Expertsysteem voor internet GIS.** Casestudy, TUDelft, Faculteit Civiele Techniek en Geowetenschappen, afdeling Geodesie, 1999.

Schans, R. van der, **GIS-cartografie.** Delft: Technische Universiteit Delft, Faculteit Civiele Techniek en Geowetenschappen, Afdeling Geodesie, 1999.

Schouten, R., **Tilburgse vastgoedinformatie via datawarehouse beschikbaar**. VIMatrix, jaargang 8, no. 6 (2000), pp. 7-10.

Thewessen, T. , B. Gehrels, **Jongste OCG specificaties veelbelovend voor geodatagebruik via internet.** VIMatrix, jaargang 8, no. 1 (2000), pp. 22-25.

Tuinman, Frank, Peter van Oosterom, **Geo-Shop, een multiserver internet-GIS**. Geodesia, no. 4 (1997), pp. 165-174.
Vçkovski, A., **Interoperable and distributed processing in GIS.** London : Taylor and Francis, 1998.

Vçkovski, A., **Interoperating geographic information systems.** Second international conference on Interoperating geographic Information Systems, INTEROP'99, Berlin : Springer, 1999.

Voss, H., N. Andrienko, G. Andrienko, **Kaartinformatie op internet vergroot klantgerichte insteek.** VIMatrix, jaargang 7, nr. X (1999), pp. 24-27.

Zhan, F.B., Barbara P. Buttenfield, **Object-oriented knowledge bases symbol selection for visualising statistical information.** International Journal of Geographical Information Systems, vol. 9, no. 3. (1995), pp. 293-315.

bezochte internetadressen

Autodesk MapGuide
http://www.mapguide.com
Descartes/Kinds
http://www.mimas.ac.uk/descartes
ESRI Map Objects
http://www.esri.com
GeoData Explorer
http://geode.usgs.gov of http://dss1.er.usgs.gov
HSL Atlas
http://www.hslzuid.nl/hsl
Magma/Lava handleiding
http://www.pgs.nl/lava/manuals/magma
Map Machine
http://plasma.nationalgeographic.com/mapmachine/plates.html
NCGI-website
http://www.ncgi.nl
Open GIS Consortium
http://www.opengis.org
Tiger Mapping
http://tiger.census.gov/cgi-bin/mapbrowse-tbl

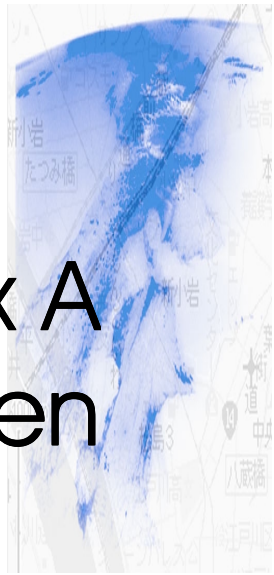
gebruikte software

Java™ 2 SDK, Standard Edition, v 1.3
http://java.sun.com/j2se/1.3/

Java™ 2 SDK, Standard Edition, v 1.2.2
http://java.sun.com/products/jdk/1.2/

JBuilder™ 3.5
http://www.borland.com/jbuilder/jb35/

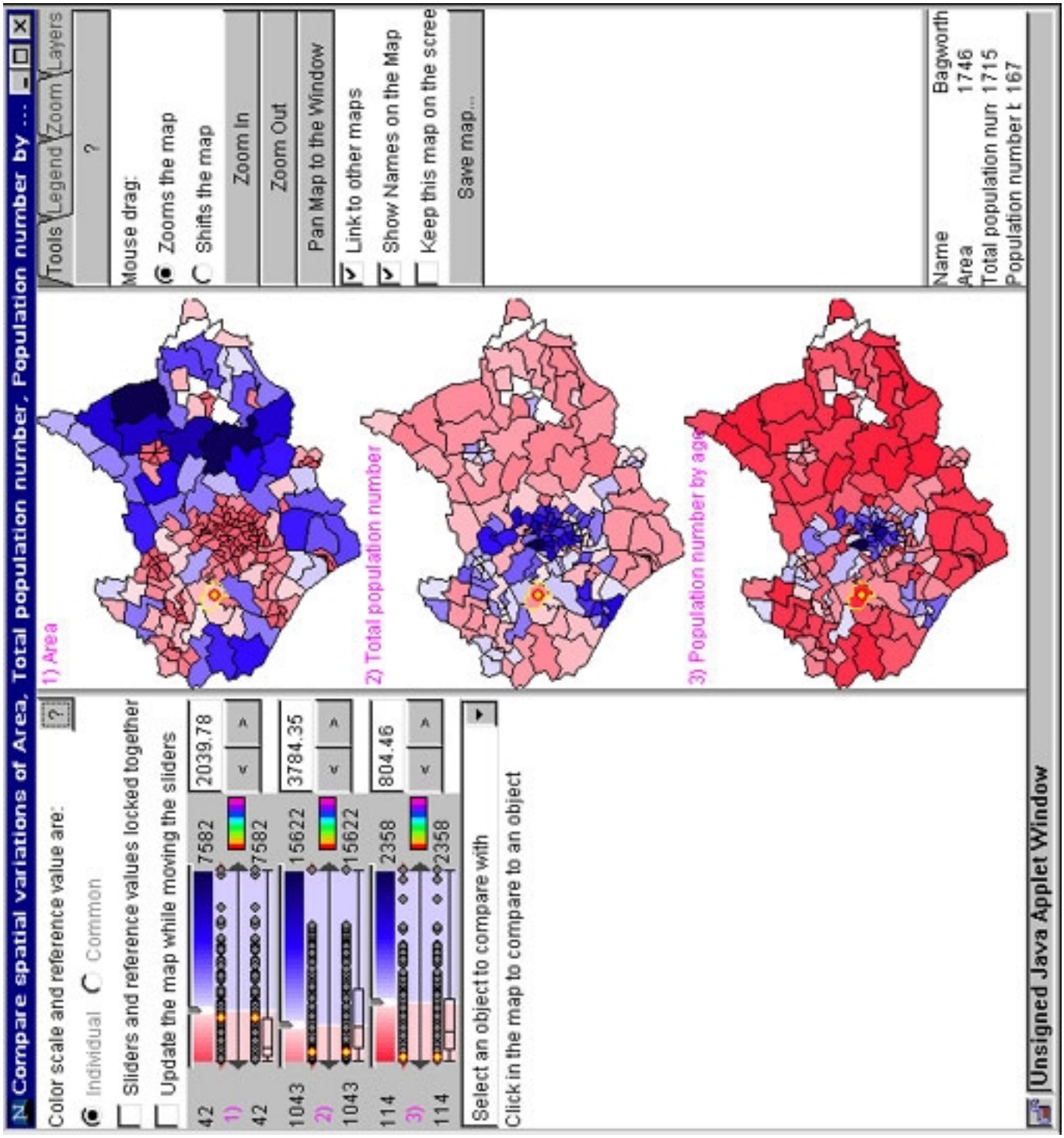
Magma/Lava
http://www.pgs.nl/lava



Appendix A
GIS atlassen

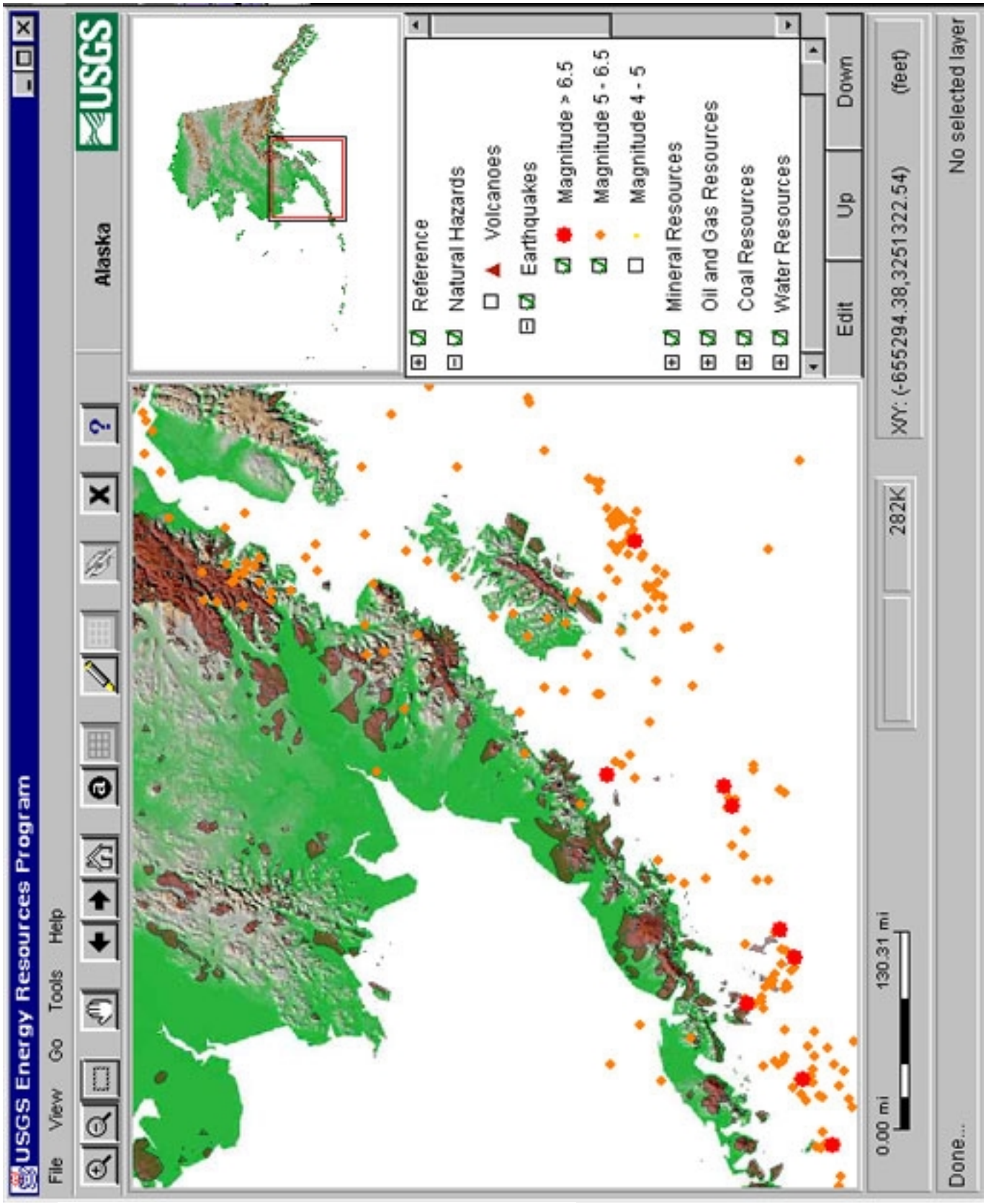
Figuur A1 Map Machine

Figuur A2 HSL Atlas

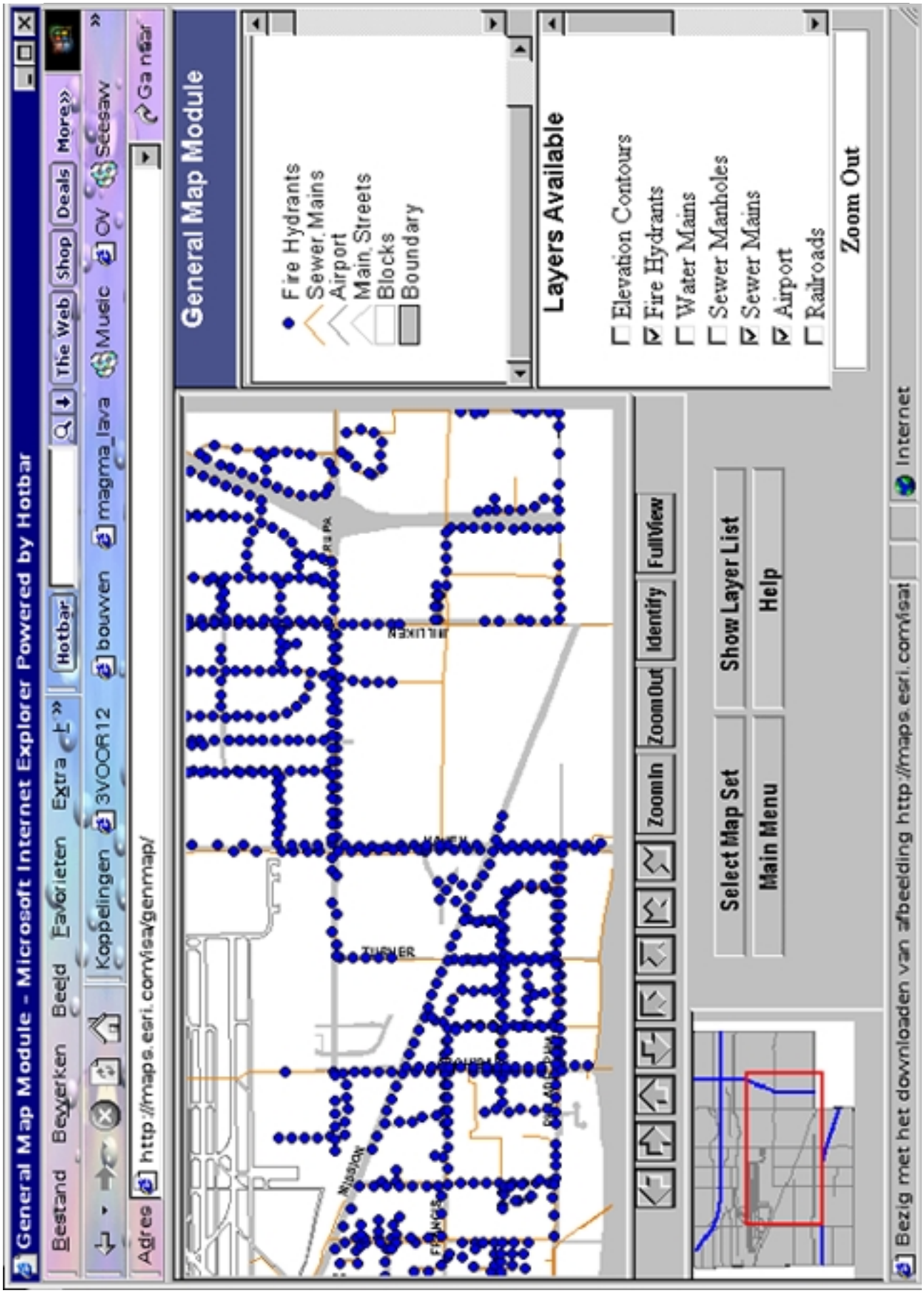


Figuur A3 Descartes/Kinds

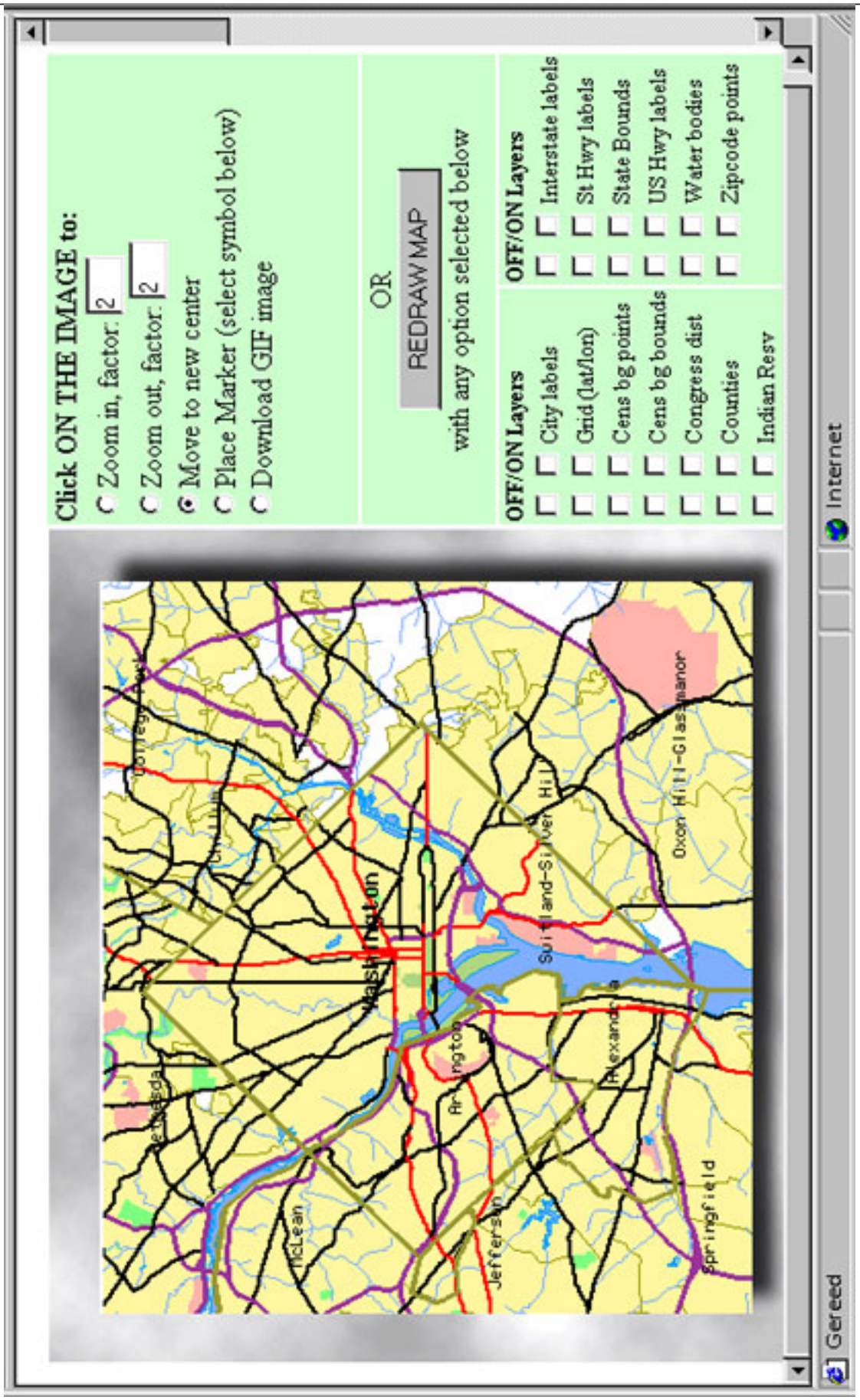
Figuur A4 Autodesk MapGuide



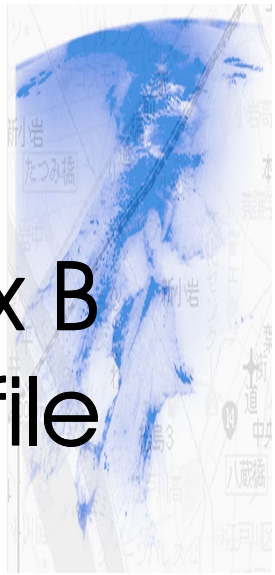
Figuur A5 Geo Data Explorer



Figuur A6 ESRI Map Objects



Figuur A7 Tiger mapping



Appendix B configuratiefile

home/export/magma/binaries/lava_dev/magma_rel.conf

```
▼
▼
//---------- Lava Options ----------
lavaOptions=
LavaOptions(
  wwwHome = "http://www.gdmc.nl/lava_dev",
  magmaCGI = "magma_rel",
  iconURL = "http://www.gdmc.nl/lava/icons",
  isDemo = false,
  showCacheShadow = false,
  showGridButtons = true,
  showDigitizeButtons = true,
  showOVZoomButtons = true,
  showLayerButtons = true,
  showThematicButton = true,
  showVisitPGS = false,
  printCorrection = 1.68,
  maxMemoryUsage = 18192000,
  frameTitle = "PGS/TU Delft Internet GIS (Lava Development, Tilburg & Kadaster data)",
  language = 2,
  debugging = false,
  debuggingRDB = false,
  useCache = true,
  menuMaps=(
    MenuMap(
      name = "BasisGeografie Tilburg",
      overView = lookup("luchtfoto"),
      overViewArea = WRectangle(x1=124000,y1=392500,x2=144500,y2=409500),
      detailView = lookup("luchtfoto"),
      detailViewArea = WRectangle(x1= 133350,y1= 395850,x2= 134750,y2= 397000)
    ),
  )
  startWithMap = lookup("BasisGeografie Tilburg")
);
//---------- Available magma servers ----------

magmaServers = List<MagmaServer>(
  MagmaServer(
    name = "www.gdmc.nl/cgi-bin"
    databases = List<Database>(
      Database(
        name = "tilburg",
        fullName = "Polygon",
        projection = "RDM",
        info = "Info Datawarehouse Tilburg"
        relations=(
          Relation(
        name="SPOORLIJNMGM",
        numTuples = 274,
            range = WRectangle(x1=124000,y1=392500,x2=141000,y2=406000)
      )
          Relation(
        name="STADSD99MGM",
        numTuples = 10,
            range = WRectangle(x1=124000,y1=392500,x2=141000,y2=406000)
      )
▼
▼
 //---------- Available maps ----------

maps = List<Map>(
  Map(
    name = "luchtfoto",   // staat bovenin genoemd. Moet veranderen.
    layers = (
      Layer(
    name = "HuisnummerAdressen",
    relation = lookup("www.gdmc.nl/cgi-bin","tilburg","HUISNUMMERSMGM"),
    lavaShapeSpec=(
          type = "PointLavaShape"
          arguments = ("HUISNUMMERSATTR")
          properties = PropertyList("color=2,Shape=4"),
          propertiesOne = PropertyList("color=2,Shape=4"),
```

```
              propertiesTwo = PropertyList("color=3,Shape=4"),
              propertiesThree = PropertyList("color=4,Shape=4"),
              attributes =
AttributeList("'prefix:','http://www.gdmc.nl:81/info/vraagadres?adres=','links:',ADRESNR,'targ
et:','main'")
            ),
            visible = false,
            scaleRange=(from=0.01,to=2.5)
            colorMap = lookup("standard")
          )
Layer(
  name = "Gemeentegrens",
  relation = lookup("www.gdmc.nl/cgi-bin","tilburg","GEMGRENS99MGM"),
  lavaShapeSpec=(
    type = "PolygonLavaShape"
    arguments = ("GEMGRENS99ATTR")
    properties = PropertyList("BorderColor=2,FillPoly=0,BorderWidth=1,color=1,pickable=0")
    propertiesOne=PropertyList("BorderColor=2,FillPoly=0,BorderWidth=1,color=1,pickable=0"),
     propertiesTwo=PropertyList("BorderColor=10,FillPoly=0,BorderWidth=1,color=1,pickable=0"),
    propertiesThree=PropertyList("BorderColor=9,FillPoly=0,BorderWidth=1,color=1,pickable=0")

              ),
            visible = false,
            colorMap = lookup("standard")
          )
          Layer(
            name = "Perceelgrenzen",
            relation = lookup("www.gdmc.nl/cgi-bin","tilburg","TESTMGM"),
            lavaShapeSpec=(
              type = "PolylineLavaShape"
              arguments = ("TESTATTR")
              properties = PropertyList("color=2,pickable=0")
              propertiesOne = PropertyList("color=2,pickable=0"),
              propertiesTwo = PropertyList("color=19,pickable=0"),
              propertiesThree = PropertyList("color=14,pickable=0")
           ),
            visible = false,
            scaleRange=(from=0.01,to=4)
            colorMap = lookup("standard")
          )
    ▼
    ▼
    //---------- Colormap ----------------

    colorMaps = List<ColorMap>(
      ColorMap(
        name = "standard",
        colors = (
            1 = Color(r=255,g=255,b=255),
            2 = Color(0,0,0),
            3 = Color(255,255,0),
            4 = Color(255,0,0),
            5 = Color(0,255,0),
            6 = Color(0,0,255),
            7 = Color(255,180,0),
            8 = Color(0,180,170),
            9 = Color(0,0,127),
           10 = Color(0,85,0),
           11 = Color(0,170,0)
           12 = Color(0,170,127),
           13 = Color(0,170,255),
    ▼
    ▼
```

Appendix C
Java specificatie,tools en Java files

de Lava klassenhiërarchie

home/export/magma/sources/lava_dev/nl/pgs/lava

```
    /browser
    /debug
    /exception
    /extra
    /geometry
    /info
    /javaui
        /texlayout
    /javashape
    /main
    /mapview
    /mscript
    /projection
    /util
    /vulcan
```

De belangrijkste packages voor de implementatie in dit afstudeeronderzoek zijn:
Browser: bevat onder andere de klasse die de browser opstart, de lijst van andere klassen doorloopt en de initiële kaart maakt, en de klasse die het hoofdframe van het applet beschrijft;
LavaShape: beschrijft een basis voor alle LavaShapes die geografische objecten representeren in een MapView;
Mapview: bevat de klassen die bepalen wat er in de kaart komt, hoe dat te zien is en hoe Lava de kaart moet tekenen. Hier staan ook de belangrijke klassen voor de legenda en de klasse die een verandering van de eigenschappen van een laag bewerkstelligt.
MScript: beschrijft de manier waarop de informatiestroom uit de configuratiefile gelezen wordt en declareert welke klassen en variabelen er allemaal in Lava en in de configuratiefile voorkomen.

ontwikkelen in Java

```
Gebruikt: Java 2 versie 1.2.0 en 1.3.0
          JBuilder versie 3
```

Na het veranderen van een file wordt er gecompileerd door het commando <make> te geven. Als er geen fouten worden gemeld, wordt er een nieuwe JAR aangemaakt door het commando <make jar>, waarna vervolgens de nieuwe JAR naar de juiste directory gezet wordt door het commando <make install>. De veranderingen zijn nu van toepassing op het applet als het opgestart wordt. De specificaties van deze <make> commando's staan in de makefile:

home/export/magma/sources/lava_dev/makefile

```
JAVAC=javac -classpath .
JAVAC_OPTS=-J-mx64M
JAR=jar

JAVAFILES=nl/pgs/lava/browser/Browser.java $(shell find . -name '*.java' |
grep -v 'Browser.java')
CLASSFILES=${JAVAFILES:.java=.class}

LAVAVERSION=$(shell basename `pwd`)
DATE=$(shell date)
```

```
    PACKAGEDIRS = $(shell find nl | grep -v '.*\..*' | grep -v '^nl$$' | grep -v
    '^nl/pgs$$')
    PACKAGENAMES = $(shell echo ${PACKAGEDIRS})

    code: ${CLASSFILES}

    jar:: version code
     rm -f Lava.jar; ${JAR} cvf Lava.jar `find nl -name '*.class'` `find
    descartes -name '*.class'` `find netscape -name '*.class'`

    %.class: %.java
     ${JAVAC} ${JAVAC_OPTS} `echo $*.java`

    clean::
     rm -f `find nl/pgs/lava -name '*.class'`

    install::
     cp Lava.jar $(HOME)/binaries/lava_dev

    doc::
     cd doc/lapi; rm -f *.html; javadoc -package ${PACKAGENAMES}

    parsers::
     cd nl/pgs/lava/mscript; javacc MScriptParser.jj

    version::
     echo ${LAVAVERSION};\
     sed \
     's/public String version = \"[^\"]*\";/public String version =
    \"${LAVAVERSION} (${DATE})\";/' \
     nl/pgs/lava/mscript/LavaOptions.java > /tmp/$$$$; \
     cp /tmp/$$$$ nl/pgs/lava/mscript/LavaOptions.java; rm /tmp/$$$$
     rm -f ../curversion
     ln -s ${LAVAVERSION} ../curversion

all:: clean version parsers code doc jar
```

veranderde javafiles

home/export/magma/sources/lava_dev/nl/pgs/lava/Browser/LT.java

```
//=======================LT.java ======================
//
// Copyright 1998, all rights reserved.
// Professional GEO Systems BV.
//
package nl.pgs.lava.browser;

import java.util.*;
import java.io.*;
import java.lang.reflect.*;
import nl.pgs.lava.info.Info;

public class LT
{

static Hashtable h= null;

public static int language=2;

static String itz[][]= {

{ "menu0", "File", "Bestand"},
{ "menu1", "Overview", "Overzicht"},
{ "menu2", "MainView", "Detailkaart"},
{ "menu3", "Debug", "Debug"},
{ "menu5", "Advanced", "Extra"},
{ "menu00", "Clear all", "Wis alles"},
{ "menu07", "Clear all cache", "Wis cache"},
{ "menu02", "About", "Info"},
{ "menu04", "Visit PGS homepage", "Ga naar PGS homepage"},
{ "menu06", "Print", "Afdrukken"},
{ "menu9", "Exit", "Afsluiten"},
{ "menu10", "Select layers", "Selecteer lagen"},
{ "menu11", "Set coordinates", "Ga naar coordinaat"},
{ "menu20", "Set coordinates", "Ga naar coordinaat"},
{ "menu21", "Go to zipcode", "Ga naar postcode"},
{ "menu22", "Go to city", "Ga naar stad"},
{ "menu23", "Set scale 1:500", "Kies schaal 1:500"},
{ "menu24", "Set scale 1:1000", "Kies schaal 1:1000"},
{ "menu25", "Set scale 1:2000", "Kies schaal 1:2000"},
{ "menu26", "Set scale 1:5000", "Kies schaal 1:5000"},
{ "menu27", "Set scale 1:10000", "Kies schaal 1:10000"},
{ "menu50", "Digitize line", "Digitaliseer lijn"},
{ "menu51", "Digitize point", "Digitaliseer punt"},
{ "menu52", "Add an highway segment to set", "Voeg snelweg toe aan selectie"},
{ "menu53", "Clear the highway set", "Nieuwe selectie"},
{ "menu54", "Show cable profile", "Toon kabel profiel"},
{ "menu55", "Add objects arund (160000,470000)", "Add (160000,470000)"},
{ "menu100", "SHOW OBJECT INFO", "TOON OBJECT INFO"},

{ "butt1", "Ok", "Ok"},
{ "butt2", "Cancel", "Annuleren"},
{ "butt3", "Edit", "Bewerk"},
{ "butt4", "Up", "Omhoog"},
{ "butt5", "Down", "Omlaag"},
{ "butt6", "Optimalise visualisation", "Optimaliseren visualisatie"},


{ "mesg1", "Initializing maps...please wait.",
  "Initialisatie kaarten...duurt even"},
{ "mesg2", "Please digitize a Point...", "Digitaliseer een punt ..."},
{ "mesg3", "Please digitize a Line...", "Digitaliseer een lijn ..."},
{ "mesg4", "Got NO line...", "GEEN lijn gedigitaliseerd ..."},
{ "mesg5", "Please digitize cross section line...",
  "Digitaliseer lijn voor doorsnede"},
{ "mesg6", "Please pick a Highway from the Raster Map...",
  "Selecteer een Highway van de Raster Kaart ..."},
{ "mesg7", "Digitized point at: (", "Punt gedigitaliseerd op: ("},
```

```
{ "mesg10", "Done", "Klaar"},
{ "mesg11", "Interrupted!", "Geinterrumpeerd!"},
{ "mesg12", "Length: ", "Lengte: "},
{ "mesg13", "Updating layer ", "Updating laag: "},
};

static void setlanguage (int l) {
    language= l;
}

public static String t (String key) {
    if (h==null) init();
    return ((String[]) h.get(key))[language];
}

static void init () {
    h= new Hashtable (itz.length);
    language= Info.lo.language;
    for (int i=0; i<itz.length; i++)
        h.put (itz[i][0], itz[i]);
}

/*
public static void main (String argv[]) {
    System.out.println (itz[1][0] + ":" + itz[1][1] + ":"  +
        itz[1][2] + " l:" + itz.length + " l1: " + itz[1].length) ;
}
*/
}
```

home/export/magma/sources/lava_dev/nl/pgs/lava/LavaShape/LavaShape.java

```
//======================== LavaShape.java ========================
//
// Copyright 1998, all rights reserved.
// Professional GEO Systems BV.
//
package nl.pgs.lava.lavashape;

▼
▼
  //-------------------- getProperties --------------------
  /**
  * Return the initial or current properties defined for this type of LavaShape.
  * A new LSProperties instance is returned.
  */
  public LSProperties getProperties()
  {
    LSProperties p = new LSProperties();
    p.addProperty("Color", 0, 2, 32);
    p.addProperty("Pickable", 0, 1, 1);
    return p;
  }
//-------------------- getPropertiesOne --------------------
  /**
  * Return the propertiesOne defined for this type of LavaShape.
  * A new LSProperties instance is returned.
  */
  public LSProperties getPropertiesOne()
  {
    LSProperties p = new LSProperties();
    p.addProperty("Color", 0, 2, 32);
    p.addProperty("Pickable", 0, 1, 1);
    return p;
  }
//-------------------- getPropertiesTwo --------------------
  /**
  * Return the propertiesTwo defined for this type of LavaShape.
  * A new LSProperties instance is returned.
  */
  public LSProperties getPropertiesTwo()
  {
    LSProperties p = new LSProperties();
    p.addProperty("Color", 0, 2, 32);
    p.addProperty("Pickable", 0, 1, 1);
    return p;
  }
  //-------------------- getPropertiesThree --------------------
  /**
  * Return the propertiesThree defined for this type of LavaShape.
  * A new LSProperties instance is returned.
  */
  public LSProperties getPropertiesThree()
  {
    LSProperties p = new LSProperties();
    p.addProperty("Color", 0, 2, 32);
    p.addProperty("Pickable", 0, 1, 1);
    return p;
  }
▼
▼
//-------------------- updateProperties ------------------
  /**
  * Called when properties have changed. This shape can change
  * it's internal state by looking at the properties.
  */
  public void updateProperties(LSProperties p)
  {
    colorIndex = p.value[0];
 pickable = (p.value[1] != 0);
  }
  //-------------------- updatePropertiesOne ------------------
  /**
  * Called when properties have changed. This shape can change
```

```
* it's internal state by looking at the properties.
*/
public void updatePropertiesOne(LSProperties p)
{
  colorIndex = p.value[0];
  pickable = (p.value[1] != 0);
}
//------------------- updatePropertiesTwo ------------------
/**
* Called when properties have changed. This shape can change
* it's internal state by looking at the properties.
*/
public void updatePropertiesTwo(LSProperties p)
{
  colorIndex = p.value[0];
  pickable = (p.value[1] != 0);
}
 //------------------- updatePropertiesThree ------------------
/**
* Called when properties have changed. This shape can change
* it's internal state by looking at the properties.
*/
public void updatePropertiesThree(LSProperties p)
{
  colorIndex = p.value[0];
  pickable = (p.value[1] != 0);
}
```

▼
▼

home/export/magma/sources/lava_dev/nl/pgs/lava/LavaShape/PolylineLavaShape.java

```
//======================= PolylineLavaShape.java ======================
//
// Copyright 1998, all rights reserved.
// Professional GEO Systems BV.
//

▼
▼
  //-------------------- getProperties --------------------
  /**
   * Return the properties defined for this type of LavaShape.
   * A new copy of LSProperties is returned.
   */
  public LSProperties getProperties()
  {
    /*
       LSProperties p = new LSProperties();
       p.addProperty("Color", 0, 2, 32);
     */
    LSProperties p = super.getProperties();
    p.addProperty("Width", 1, 1, 10);
    p.addProperty("Pattern", 0, 0, 2);
    return p;
  }
//-------------------- getPropertiesOne --------------------
  /**
   * Return the propertiesOne defined for this type of LavaShape.
   * A new copy of LSProperties is returned.
   */
  public LSProperties getPropertiesOne()
  {
    /*
       LSProperties p = new LSProperties();
       p.addProperty("Color", 0, 2, 32);
     */
    LSProperties p = super.getPropertiesOne();
    p.addProperty("Width", 1, 1, 10);
    p.addProperty("Pattern", 0, 0, 2);
    return p;
  }
 //-------------------- getPropertiesTwo --------------------
  /**
   * Return the properties defined for this type of LavaShape.
   * A new copy of LSProperties is returned.
   */
  public LSProperties getPropertiesTwo()
  {
    /*
       LSProperties p = new LSProperties();
       p.addProperty("Color", 0, 2, 32);
     */
    LSProperties p = super.getPropertiesTwo();
    p.addProperty("Width", 1, 1, 10);
    p.addProperty("Pattern", 0, 0, 2);
    return p;
  }
//-------------------- getPropertiesThree --------------------
  /**
   * Return the propertiesThree defined for this type of LavaShape.
   * A new copy of LSProperties is returned.
   */
  public LSProperties getPropertiesThree()
  {
    /*
       LSProperties p = new LSProperties();
       p.addProperty("Color", 0, 2, 32);
     */
```

```
    LSProperties p = super.getPropertiesThree();
    p.addProperty("Width", 1, 1, 10);
    p.addProperty("Pattern", 0, 0, 2);
    return p;
  }
▼
▼
//------------------- updateProperties ------------------
  public void updateProperties(LSProperties p)
  {
    if (!useOwnColor) {
      colorIndex = p.value[0];
    }
    pickable = (p.value[1] != 0);
    if (!useOwnWidth) {
      width = (short ) p.value[super.nrProps];
    }
    if (!useOwnPattern) {
      setPattern(p.value[super.nrProps + 1]);
    }
  }
  //------------------- updatePropertiesOne ------------------
  public void updatePropertiesOne(LSProperties p)
  {
    if (!useOwnColor) {
      colorIndex = p.value[0];
    }
    pickable = (p.value[1] != 0);
    if (!useOwnWidth) {
      width = (short ) p.value[super.nrProps];
    }
    if (!useOwnPattern) {
      setPattern(p.value[super.nrProps + 1]);
    }
  }
 //------------------- updatePropertiesTwo ------------------
  public void updatePropertiesTwo(LSProperties p)
  {
    if (!useOwnColor) {
      colorIndex = p.value[0];
    }
    pickable = (p.value[1] != 0);
    if (!useOwnWidth) {
      width = (short ) p.value[super.nrProps];
    }
    if (!useOwnPattern) {
      setPattern(p.value[super.nrProps + 1]);
    }
  }
 //------------------- updatePropertiesThree ------------------
  public void updatePropertiesThree(LSProperties p)
  {
    if (!useOwnColor) {
      colorIndex = p.value[0];
    }
    pickable = (p.value[1] != 0);
    if (!useOwnWidth) {
      width = (short ) p.value[super.nrProps];
    }
    if (!useOwnPattern) {
      setPattern(p.value[super.nrProps + 1]);
    }
  }

}
```

home/export/magma/sources/lava_dev/nl/pgs/lava/Mapview/Layer.java

```
//============================ Layer.java ============================
//
// Copyright 1998, all rights reserved.
// Professional GEO Systems BV.
//
▼
▼

  public LSProperties properties = new LSProperties();
  public LSProperties propertiesOne = new LSProperties();
  public LSProperties propertiesTwo = new LSProperties();
  public LSProperties propertiesThree = new LSProperties();
  public int prior;

▼
▼
   // Create the properties associated with this layer.
      createProperties();
      if (lavaShapeSpec.properties != null)
    properties.parse(lavaShapeSpec.properties.string);    //TO BE CHANGED!

   // Create the propertiesOne associated with this layer.
          createPropertiesOne();
      if (lavaShapeSpec.propertiesOne != null)
    propertiesOne.parse(lavaShapeSpec.propertiesOne.string);  //added

   // Create the propertiesTwo associated with this layer.
          createPropertiesTwo();
      if (lavaShapeSpec.propertiesTwo != null)
    propertiesTwo.parse(lavaShapeSpec.propertiesTwo.string);  //added

   // Create the propertiesThree associated with this layer.
          createPropertiesThree();
      if (lavaShapeSpec.propertiesThree != null)
    propertiesThree.parse(lavaShapeSpec.propertiesThree.string);  //added

    }
    else {
      withinScale = true;
      layerValid = true;
      layerComplete = false;
    }
▼
▼

  //-------------------- createProperties --------------------
  /**
   * Create properties for this layer, depending on the
   * LavaShape type contained in this layer.
   */
  public void createProperties()
  {
    if (view == null) {
      Debug.println("createProperties(): no view!");
      return ;
    }
    String type = lavaShapeAttr.type.intern();

    // REFLECTION
    if (JDKVersion.JDK11_reflection && false) {
      // JDK 1.1 a.l.
      try {
    Class cl = Class.forName("nl.pgs.lava.lavashape." + type);
    Method methd = cl.getMethod("getProperties", null);
    properties = (LSProperties) methd.invoke(null, null);
      }
      catch (Exception e) {
```

```
      Debug.panic();
        }
      }
      else {
        // JDK 1.0.2
        try {
      Class cl = Class.forName("nl.pgs.lava.lavashape." + type);
      Object inst = cl.newInstance();
      properties = ((LavaShape) inst).getProperties();
        }
        catch (Exception e) {
      Debug.panic();
        }
      }
    }
    //-------------------- createPropertiesOne --------------------
    /**
     * Create propertiesOne for this layer, depending on the
     * LavaShape type contained in this layer.
     */
    public void createPropertiesOne()
    {
      if (view == null) {
        Debug.println("createPropertiesOne(): no view!");
        return ;
      }
      String type = lavaShapeAttr.type.intern();

      // REFLECTION
      if (JDKVersion.JDK11_reflection && false) {
        // JDK 1.1 a.l.
        try {
      Class cl = Class.forName("nl.pgs.lava.lavashape." + type);
      Method methd = cl.getMethod("getPropertiesOne", null);
      propertiesOne = (LSProperties) methd.invoke(null, null);
        }
        catch (Exception e) {
      Debug.panic();
        }
      }
      else {
        // JDK 1.0.2
        try {
      Class cl = Class.forName("nl.pgs.lava.lavashape." + type);
      Object inst = cl.newInstance();
      propertiesOne = ((LavaShape) inst).getPropertiesOne();
        }
        catch (Exception e) {
      Debug.panic();
        }
      }
    }

    //-------------------- createPropertiesTwo --------------------
    /**
     * Create propertiesTwo for this layer, depending on the
     * LavaShape type contained in this layer.
     */
    public void createPropertiesTwo()
    {
      if (view == null) {
        Debug.println("createPropertiesTwo(): no view!");
        return ;
      }
      String type = lavaShapeAttr.type.intern();

      // REFLECTION
      if (JDKVersion.JDK11_reflection && false) {
        // JDK 1.1 a.l.
```

```
        try {
      Class cl = Class.forName("nl.pgs.lava.lavashape." + type);
      Method methd = cl.getMethod("getPropertiesTwo", null);
      propertiesTwo = (LSProperties) methd.invoke(null, null);
        }
        catch (Exception e) {
      Debug.panic();
        }
      }
      else {
        // JDK 1.0.2
        try {
      Class cl = Class.forName("nl.pgs.lava.lavashape." + type);
      Object inst = cl.newInstance();
      propertiesTwo = ((LavaShape) inst).getPropertiesTwo();
        }
        catch (Exception e) {
      Debug.panic();
        }
      }
    }

   //-------------------- createPropertiesThree --------------------
    /**
     * Create propertiesThree for this layer, depending on the
     * LavaShape type contained in this layer.
     */
    public void createPropertiesThree()
    {
      if (view == null) {
        Debug.println("createPropertiesThree(): no view!");
        return ;
      }
      String type = lavaShapeAttr.type.intern();

      // REFLECTION
      if (JDKVersion.JDK11_reflection && false) {
        // JDK 1.1 a.l.
        try {
      Class cl = Class.forName("nl.pgs.lava.lavashape." + type);
      Method methd = cl.getMethod("getPropertiesThree", null);
      propertiesThree = (LSProperties) methd.invoke(null, null);
        }
        catch (Exception e) {
      Debug.panic();
        }
      }
      else {
        // JDK 1.0.2
        try {
      Class cl = Class.forName("nl.pgs.lava.lavashape." + type);
      Object inst = cl.newInstance();
      propertiesThree = ((LavaShape) inst).getPropertiesThree();
        }
        catch (Exception e) {
      Debug.panic();
        }
      }
    }
▼
▼
   //-------------------- getProperties --------------------
     /**
      * Get the LSProperties list belonging to this layer.
      */
     public LSProperties getProperties()
     {
       return properties;
     }
```

```
//-------------------- getPropertiesOne --------------------
  /**
   * Get the LSProperties list belonging to this layer.
   */
  public LSProperties getPropertiesOne()
  {
    return propertiesOne;
  }
  //-------------------- getPropertiesTwo --------------------
  /**
   * Get the LSProperties list belonging to this layer.
   */
  public LSProperties getPropertiesTwo()
  {
    return propertiesTwo;
  }
//-------------------- getPropertiesThree --------------------
  /**
   * Get the LSProperties list belonging to this layer.
   */
  public LSProperties getPropertiesThree()
  {
    return propertiesThree;
  }
▼
▼

  //-------------------- updateProperties --------------------
  /**
   * Inform the layer that it's properties have changed.
   * This information has to be propagated to all the LavaShapes.
   */
  protected void updateProperties()
  {
    if (view == null)
      return;
    LavaShape cur_ls;
    if (type == DUMMY || type == PARENT)
      return;

    Enumeration enum = lavaShapeAttr.valueEnumeration();
    while (enum.hasMoreElements()) {
      cur_ls = (LavaShape) enum.nextElement();
      cur_ls.updateProperties(properties);
      cur_ls.setLayer(this);
    }
  }
  //-------------------- updatePropertiesOne --------------------
  /**
   * Inform the layer that it's properties have changed.
   * This information has to be propagated to all the LavaShapes.
   */
  protected void updatePropertiesOne()
  {
    if (view == null)
      return;
    LavaShape cur_ls;
    if (type == DUMMY || type == PARENT)
      return;

    Enumeration enum = lavaShapeAttr.valueEnumeration();
    while (enum.hasMoreElements()) {
      cur_ls = (LavaShape) enum.nextElement();
      cur_ls.updatePropertiesOne(propertiesOne);
      cur_ls.setLayer(this);
    }
  }
 //-------------------- updatePropertiesTwo --------------------
  /**
```

```
     * Inform the layer that it's properties have changed.
     * This information has to be propagated to all the LavaShapes.
     */
    protected void updatePropertiesTwo()
    {
      if (view == null)
        return;
      LavaShape cur_ls;
      if (type == DUMMY || type == PARENT)
        return;

      Enumeration enum = lavaShapeAttr.valueEnumeration();
      while (enum.hasMoreElements()) {
        cur_ls = (LavaShape) enum.nextElement();
        cur_ls.updatePropertiesTwo(propertiesTwo);
        cur_ls.setLayer(this);
      }
    }
    //-------------------- updatePropertiesThree --------------------
    /**
     * Inform the layer that it's properties have changed.
     * This information has to be propagated to all the LavaShapes.
     */
    protected void updatePropertiesThree()
    {
      if (view == null)
        return;
      LavaShape cur_ls;
      if (type == DUMMY || type == PARENT)
        return;

      Enumeration enum = lavaShapeAttr.valueEnumeration();
      while (enum.hasMoreElements()) {
        cur_ls = (LavaShape) enum.nextElement();
        cur_ls.updatePropertiesThree(propertiesThree);
        cur_ls.setLayer(this);
      }
    }
▼
▼
```

home/export/magma/sources/lava_dev/nl/pgs/lava/Mapview/Legend.java

```
//====================== Legend.java ======================
//

public class Legend extends Panel implements
  ItemListener,      // listen to rows
  ActionListener,    // row.edit
  DialogAgent
{
  private MapView mapView;
  private LegendRow selectedRow;
  private Layer selectedLayer;
  private GridBagLayout gbl;
  private GridBagConstraints gbc;

  private static final int PAINTONLY = 1;
  private static final int REBUILD = 2;
  private int repaintAction = PAINTONLY;

  Vector rows;   // list with all rows (LegendRow)

  AScrollPane scrollPane;
  Panel legendPanel;
  Button editButton;
  Button upButton;
  Button downButton;
  Button optimalButton;
▼
▼
    optimalButton = new Button(LT.t("butt6"));
    optimalButton.addActionListener(this);
    gbc.gridx = 0;
    gbc.gridy = 1;
    gbc.gridwidth = GridBagConstraints.REMAINDER;
    gbc.gridheight = 1;
    gbc.fill = GridBagConstraints.BOTH;
    gbc.anchor = GridBagConstraints.NORTH;
    gbc.weightx = 0.0;
    gbc.weighty = 0.0;
    pGBL.setConstraints(optimalButton, gbc);
    add(optimalButton);

▼
▼

//:::::::::::::::::::: ActionListener ::::::::::::::::::::

  public void actionPerformed(ActionEvent e)
  {
    if (e.getSource() instanceof LegendRow) {
      LegendRow row = (LegendRow)e.getSource();

if (e.getSource() == optimalButton) {
   if (selectedRow != null) {
    Layer selectedLayer = selectedRow.layer;
        mapView.getMap().optimalProperties(selectedRow.layer);
        rebuild();
        setSelectedLayer(selectedLayer);
        mapView.repaintMap(MapView.REFRESH);
     }
▼
▼
```

home/export/magma/sources/lava_dev/nl/pgs/lava/Mapview/Map.java

```
//============================ Map.java ============================

▼
▼
private boolean active = false;
  public MapView mapView;        // mapView if map is displayed by it
  public int prior;
  public int colorOne;
  public int colorTwo;
  public int colorThree;
  public int colorFour;
  public int colorFive;
  public int colorSix;
  public int colorSeven;
  public int colorEight;

▼
▼

  //---------optimalProperties--------------------
  /**Testmethod for assigning optimal properties to the layers
  * is based on the assinged priority and the colors
  * of the layers with a higher priority
  */
  public boolean optimalProperties(Layer layer) {
       setPriorities (layer);

     if (layer.prior == 1){
           layer.properties = layer.propertiesOne;
           colorOne = layer.properties.value[0];

       }

      if (layer.prior == 2){{
           layer.properties = layer.propertiesOne;
           colorTwo = layer.properties.value[0];
            }
           if (colorTwo == colorOne) {
            layer.properties = layer.propertiesTwo;
       int colorTwo = layer.properties.value[0];
              }

      }
       if (layer.prior == 3){{
           layer.properties = layer.propertiesOne;
           colorThree = layer.properties.value[0];
                       }
           if ((colorThree == colorOne) || (colorThree == colorTwo)) {
            layer.properties = layer.propertiesTwo;
        colorThree = layer.properties.value[0];
           }
           if ((colorThree == colorOne) || (colorThree == colorTwo)) {
            layer.properties = layer.propertiesThree;
        colorThree = layer.properties.value[0];
           }

       }
    if (layer.prior == 4){{
           layer.properties = layer.propertiesOne;
           colorFour = layer.properties.value[0];
                       }
    if ((colorFour == colorOne) || (colorFour == colorTwo) || (colorFour == colorThree)) {
            layer.properties = layer.propertiesTwo;
        colorFour = layer.properties.value[0];
           }
    if ((colorFour == colorOne) || (colorFour == colorTwo) || (colorFour == colorThree)) {
            layer.properties = layer.propertiesThree;
```

```
            colorFour = layer.properties.value[0];
               }

            }
        if (layer.prior == 5){{
               layer.properties = layer.propertiesOne;
               colorFive = layer.properties.value[0];
                           }
        if ((colorFive == colorOne) || (colorFive == colorTwo) || (colorFive == colorThree)) {
                layer.properties = layer.propertiesTwo;
            colorFive = layer.properties.value[0];
               }
        if ((colorFive == colorOne) || (colorFive == colorTwo) || (colorFive == colorThree)) {
                layer.properties = layer.propertiesThree;
            colorFive = layer.properties.value[0];
               }

            }
        if (layer.prior == 6){{
               layer.properties = layer.propertiesOne;
               colorSix = layer.properties.value[0];
                           }
        if ((colorSix == colorOne) || (colorSix == colorTwo) || (colorSix == colorThree)) {
                layer.properties = layer.propertiesTwo;
            colorSix = layer.properties.value[0];
               }
        if ((colorSix == colorOne) || (colorSix == colorTwo) || (colorSix == colorThree)) {
                layer.properties = layer.propertiesThree;
            colorSix = layer.properties.value[0];
               }

            }
        if (layer.prior == 7){{
               layer.properties = layer.propertiesOne;
               colorSeven = layer.properties.value[0];
                           }
        if ((colorSeven == colorOne) || (colorSeven == colorTwo) || (colorSeven ==
colorThree)) {
                layer.properties = layer.propertiesTwo;
            colorSeven = layer.properties.value[0];
               }
        if ((colorSeven == colorOne) || (colorSeven == colorTwo) || (colorSeven ==
colorThree)) {
                layer.properties = layer.propertiesThree;
            colorSeven = layer.properties.value[0];
               }

            }
        if (layer.prior == 8){{
               layer.properties = layer.propertiesOne;
               colorEight = layer.properties.value[0];
                           }
        if ((colorEight == colorOne) || (colorEight == colorTwo) || (colorEight ==
colorThree)) {
                layer.properties = layer.propertiesTwo;
            colorEight = layer.properties.value[0];
               }
        if ((colorEight == colorOne) || (colorEight == colorTwo) || (colorEight ==
colorThree)) {
                layer.properties = layer.propertiesThree;
            colorEight = layer.properties.value[0];
               }

            }
        return true;
      }
      //-----------------setPriorities-------------
      /* Testmethod for assigning a priority to the selected layer
      * Is based on the index of the legend and invoked
```

```
* when the optimalise button is pressed
*/
public boolean setPriorities(Layer l)
  {
    Vector layerVec = findLayerVector(l);
    if (layerVec == null)
      return false;

   int index = layerVec.indexOf(l);
   l.prior = index + 1;


  return true;
  }
```

▼
▼

home/export/magma/sources/lava_dev/nl/pgs/lava/MScript/LavaShapeSpec.java

```
//============================ LavaShapeSpec.java ============================
//
▼
▼
public class LavaShapeSpec
{
  public String type = null;
  public Vector arguments = new Vector();     // Vector<String>
  public PropertyList properties = null;        // initial settings, same as first
  public PropertyList propertiesOne = null;         // First preference
  public PropertyList propertiesTwo = null;   //Added for second preference
  public PropertyList propertiesThree = null; //Added for third preference
  public AttributeList attributes = null;

  //-------------------- infoCopy --------------------
  public LavaShapeSpec infoCopy()
  {
    LavaShapeSpec l = new LavaShapeSpec();
    if (type != null)
      l.type = new String(type);
    Enumeration enum = arguments.elements();
    while (enum.hasMoreElements()) {
      String s = (String)enum.nextElement();
      l.arguments.addElement(new String(s));
    }

    if (properties != null)
      l.properties = properties.infoCopy();
    if (propertiesOne != null)
      l.propertiesOne = propertiesOne.infoCopy();
    if (propertiesTwo != null)
      l.propertiesTwo = propertiesTwo.infoCopy();
    if (propertiesThree != null)
      l.propertiesThree = propertiesThree.infoCopy();
    if (attributes != null)
      l.attributes = attributes.infoCopy();
    return l;
  }


}
```

home/export/magma/sources/lava_dev/nl/pgs/lava/MScript/MScriptParser.java

```
//============================ MScriptParser.java ============================
//

/* Generated By:JavaCC: Do not edit this line. MScriptParser.java */
package nl.pgs.lava.mscript;
▼
▼
//-------------------- LavaShapeSpec --------------------

final public LavaShapeSpec LavaShapeSpec() throws ParseException {
                                  LavaShapeSpec r = new LavaShapeSpec();
    if (jj_2_82(2)) {
      jj_consume_token(LavaShapeSpec);
    } else {
      ;
    }
    jj_consume_token(186);
    label_8:
    while (true) {
      if (jj_2_83(2)) {
        ;
      } else {
        break label_8;
      }
      if (jj_2_84(2)) {
        jj_consume_token(type);
        jj_consume_token(183);
        r.type = String();
      } else if (jj_2_85(2)) {
        jj_consume_token(arguments);
        jj_consume_token(183);
        r.arguments = List_String();
      } else if (jj_2_86(2)) {
        jj_consume_token(properties);
        jj_consume_token(183);
        r.properties = PropertyList();
      } else if (jj_2_87(2)) {
        jj_consume_token(propertiesOne);
        jj_consume_token(183);
        r.propertiesOne = PropertyList();
      } else if (jj_2_88(2)) {
        jj_consume_token(propertiesTwo);
        jj_consume_token(183);
        r.propertiesTwo = PropertyList();
      } else if (jj_2_89(2)) {
        jj_consume_token(propertiesThree);
        jj_consume_token(183);
        r.propertiesThree = PropertyList();
      } else if (jj_2_90(2)) {
        jj_consume_token(attributes);
        jj_consume_token(183);
        r.attributes = AttributeList();
      } else {
        jj_consume_token(-1);
        throw new ParseException();
      }
      if (jj_2_91(2)) {
       jj_consume_token(187);
      } else {
        ;
      }
    }
    jj_consume_token(188);
       {if (true) return r;}
    throw new Error("Missing return statement in function");
  }
▼
```

▼

```
final private boolean jj_3_89() {
    if (jj_scan_token(propertiesThree)) return true;
    if (jj_la == 0 && jj_scanpos == jj_lastpos) return false;
    if (jj_scan_token(183)) return true;
    if (jj_la == 0 && jj_scanpos == jj_lastpos) return false;
    return false;
  }
final private boolean jj_3_88() {
    if (jj_scan_token(propertiesTwo)) return true;
    if (jj_la == 0 && jj_scanpos == jj_lastpos) return false;
    if (jj_scan_token(183)) return true;
    if (jj_la == 0 && jj_scanpos == jj_lastpos) return false;
    return false;
  }
final private boolean jj_3_87() {
    if (jj_scan_token(propertiesOne)) return true;
    if (jj_la == 0 && jj_scanpos == jj_lastpos) return false;
    if (jj_scan_token(183)) return true;
    if (jj_la == 0 && jj_scanpos == jj_lastpos) return false;
    return false;
  }
final private boolean jj_3_86() {
    if (jj_scan_token(properties)) return true;
    if (jj_la == 0 && jj_scanpos == jj_lastpos) return false;
    if (jj_scan_token(183)) return true;
    if (jj_la == 0 && jj_scanpos == jj_lastpos) return false;
    return false;
  }
```

▼

▼

home/export/magma/sources/lava_dev/nl/pgs/lava/MScript/MScriptParserConstants.java

```
//============================ MScriptParserConstants.java ============================

//* Generated By:JavaCC: Do not edit this line. MScriptParserConstants.java */
package nl.pgs.lava.mscript;

public interface MScriptParserConstants {
▼
▼
  int type = 79;
  int arguments = 80;
  int properties = 81;
  int propertiesOne = 82;
  int propertiesTwo = 83;
  int propertiesThree = 84;
  int attributes = 85;
▼
▼
    "\"type\"",
    "\"arguments\"",
    "\"properties\"",
    "\"propertiesOne\"",
    "\"propertiesTwo\"",
    "\"propertiesThree\"",
    "\"attributes\"",
▼
▼
```

home/export/magma/sources/lava_dev/nl/pgs/lava/MScrip/MScriptParser.jj

```
//============================== MScriptParser.jj ==============================
//
// Copyright 1997, all rights reserved.
// Professional GEO Systems BV.
//
▼
▼
  <type:     "type"> |
  <arguments:        "arguments"> |
  <properties:           "properties"> |
  <propertiesOne:    "propertiesOne"> |
  <propertiesTwo:    "propertiesTwo"> |
  <propertiesThree:  "propertiesThree"> |
  <attributes:       "attributes"> |
▼
▼
//-------------------- LavaShapeSpec --------------------
LavaShapeSpec LavaShapeSpec() : {LavaShapeSpec r = new LavaShapeSpec();}
{
  (<LavaShapeSpec>)? "(" (
  (
    (<type>          "=" r.type=String()      )|
    (<arguments>     "=" r.arguments=List_String()    )|
    (<properties>        "=" r.properties=PropertyList()  )|
    (<propertiesOne>     "=" r.propertiesOne=PropertyList()   )|
    (<propertiesTwo>     "=" r.propertiesTwo=PropertyList()   )|
    (<propertiesThree>   "=" r.propertiesThree=PropertyList() )|
    (<attributes>        "=" r.attributes=AttributeList())
  ) (",")?
  )*
  ")" {return r;}
}
▼
▼
```