Modelleren van constraints in GIS

Een constraint is een geldigheidsvoorwaarde in het model waaraan de inhoud van de dataset altijd moet voldoen om correct te zijn. Constraints zijn vooral van belang in situaties waarbij gegevens veranderen. Constraints zijn belangrijke onderdelen van elk modelleerproces maar krijgen daarin relatief weinig aandacht. In de praktijk worden ad hoc oplossingen gebruikt die van het toepassingsdomein en de gebruikte tools afhangen.

Om de betekenis van de gegevens beter vast te leggen, moeten de constraints in het model zijn opgenomen naast de standaard-modelonderdelen als de objectklassen, attributen en relaties (specialisaties, aggregaties en associaties). Om de kwaliteit van de gegevens te kunnen garanderen, moeten de constraints in het systeem worden afgedwongen, zowel in de edit-omgeving als in de database als bij de gegevensuitwisseling. Idealiter zou de constraintondersteuning in al deze subsystemen automatisch moeten worden afgeleid uit het model.

Prof.dr.ir. P.J.M. van Oosterom, sectie GIS-Technologie, TU Delft

Op dit moment zijn er al wel een aantal deeloplossingen voor de ondersteuning van een aantal constraint-typen. Domeinwaarde-constraints en referentiële integriteit zijn standaardfunctionaliteiten in relationele databases [Date and Darwen, 1997]. Als er dus vanuit een tabel naar een object in een andere tabel wordt verwezen dan controleert de database of de verwijzing daadwerkelijk bestaat (anders zal de transactie niet doorgaan en blijft de database in de oude staat). Een ander voorbeeld van constraint-support is te vinden in de GIS-context: het afdwingen van correcte topologische relaties (bijvoorbeeld twee percelen mogen elkaar niet overlappen). Ondersteuning voor deze functionaliteit is tegenwoordig te vinden in bijvoorbeeld LaserScan Radius topology en Oracle spatial 10g of wordt ondersteund in edit- en middleware-producten als ESRI ArcGIS. Hoewel dit stappen in de goede richting zijn, is dit nog maar een gedeeltelijke oplossing want niet slechts een deelsysteem zou bepaalde constraints moeten ondersteunen, maar alle deelsystemen zouden dezelfde complete set aan constraints moeten ondersteunen afgeleid uit het model. In dit artikel wordt de noodzaak hiervan en de wijze waarop deze visie gerealiseerd zou moeten worden verder beschreven. Eerst zullen vier cases de revue passeren waaruit een grote behoefte blijkt aan (verschillende soorten) constraints. Vervolgens zal een indeling van de verschillende typen constraints worden gepresenteerd (taxonomie). Daarna zal worden aangegeven hoe constraints formeel kunnen worden gespecificeerd in een model en hoe dit model vervolgens kan worden geïmplementeerd. Ten slotte zullen de belangrijkste conclusies worden samengevat en een aantal punten voor toekomstig onderzoek worden beschreven.

Case 1: landschapsontwerpomgeving SALIX-2

De interactieve gebruikers van SALIX-2 [Van Lammeren et al, 2002] kunnen in een Virtual Reality (VR) omgeving nieuwe objecten, zoals bomen en struiken, planten in een 3D-landschap (fig. 1). Net zoals in de werkelijkheid moeten deze objecten soms een bepaalde afstand tot elkaar hebben (bijv. twee bomen mogen niet dichter dan drie meter bij elkaar staan), vormen bepaalde objecten samen een complexer object (bijv. een aantal bomen in een zeker patroon vormt een landschapsarchitectuurele-

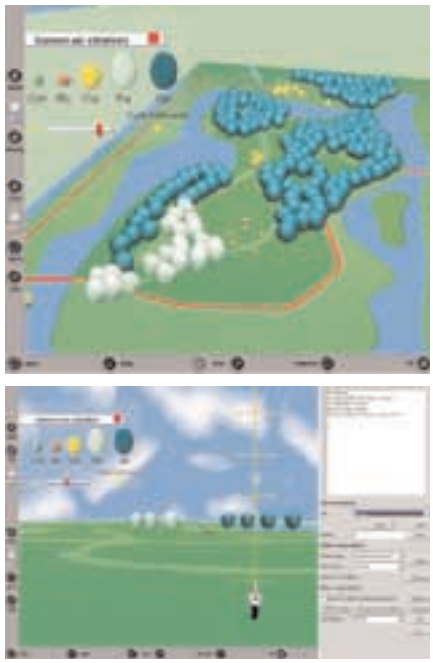

Fig. 1. De 3D interface van SALIX-2: een interactief landschapsmodelleringssysteem.

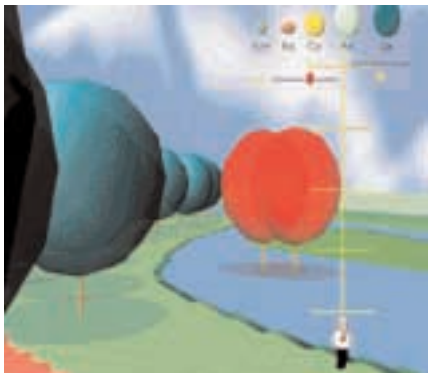

Fig. 2. Een constraint wordt overtreden in SALIX-2c (let op de rode, 'highlighted' boom).

ment) of mogen helemaal niet worden geplant in een bepaald gebied (boom mag niet op de weg staan). Naast het ontwerpen van een beplantingsplan, bestaat in SALIX-2 ook de mogelijkheid om de groei van de bomen en struiken in de tijd te simuleren. Zo kan het zijn dat bij het planten van de objecten er nog geen conflicten zijn maar dat na vijftien jaar groeien, deze wel ontstaan (fig. 2). Tabel 1 geeft een aantal voorbeelden van constraints toegevoegd aan SALIX-2 in het afstudeerwerk van Jildou Louwsma [Louwsma, 2004].

Voorbeelden van relatie constraints in SALIX-2 [Louwsma et al, 2005]

Type relatie	Constraints
Richting	Een struik moet altijd ten zuiden van boom staan
Topologie	Struiken mogen nooit in het water of op de paden staan
Metriek	Bomen moeten minimaal 1 meter van de paden staan
Temporeel	Eik moet minimaal 20 jaar vrij kunnen groeien
Kwantiteit/ Aggregatie	In gegeven specifiek gebied moeten minimaal 10 bomen staan
Thematisch	De kleur van bepaalde struiken moet voorkomen in gegeven lijst
Complex	Bomen die in het water staat moeten ten minste 8 meter uit elkaar staan EN de afstand van de boom tot de rand van het water mag niet meer zijn dan 0,5 meter EN de boomsoort moet in dit geval 'salix' zijn

Case 2: kadastrale bijhouding

De kadastrale bijhouding is per definitie een dynamisch GIS en om de gegevenskwaliteit te bewaken, zouden constraints een belangrijke rol moeten spelen; ook in relatie tot de juridisch-administratieve gegevens. De kadastrale kaart in LKI is gebaseerd op de winged-edge (ketting) topologiestructuur beheerd in een database DBMS (fig. 3). Deze dataset is, topologisch gezien, zeer 'clean'. De topologische constraints zijn hard gecodeerd in de edit-omgeving en in het database check-in proces maar niet in de database zelf. Om de kwaliteit van de kadastrale gegevens te controleren,

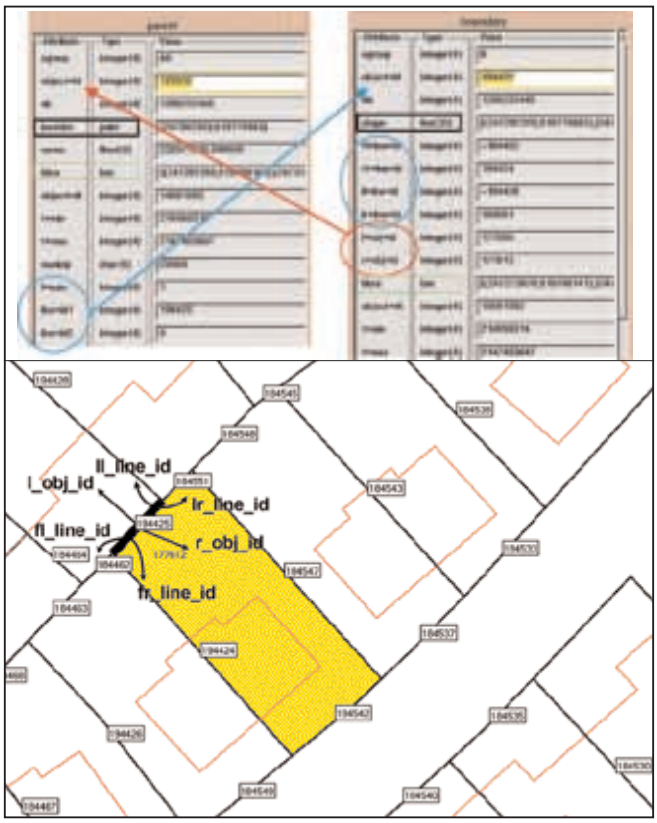

Fig. 3. Winged-edge topologie structuur van de kadastrale kaart.

zijn meer dan vijftig controles (als 'SQL select queries') ontwikkeld die in de toekomst ook als constraint in de database zouden moeten worden opgenomen. Naast de bijna triviale domeinwaarde-constraints, kunnen de ontwikkelde constraints in de volgende categorieën worden ingedeeld:

- metrische constraints;
- topologische-verwijzingen-existentie-constraints;
- topologische-verwijzingen-correctheid-constraints;
- andere referentiële-integriteits-constraints;
- temporele constraints.

Enkele voorbeelden van een *metrische constraint* zijn:

- controle of cirkelbogen zijn gesloten (eerste en laatste punt zijn identiek, hierdoor is oriëntatie van cirkel niet gedefinieerd en heeft links en rechts geen betekenis);
- een cirkelboog mag niet recht zijn;
- het perceel-referentiepunt moet binnen de omhullende rechthoek van een perceel liggen;
- twee grenzen mogen elkaar niet kruisen (alleen raken in eindpunten).

De *topologische-verwijzingen-existentie-constraints* controleren of topologische verwijzingen ook echt bestaan. Dit lijkt veel op de bekende referentiële integriteit in administratieve systemen, ware het niet dat de verwijzingen gericht zijn (van + of – teken voorzien om grens juiste richting te geven).

Enkele voorbeelden:
- controle of de winge-edge grens-grens verwijzingen bestaan;
- bestaan, vanuit een grens, de links-en rechts-verwijzingen naar percelen;
- bestaat de verwijzing vanuit een perceel naar de buitengrens;
- bestaan de verwijzingen naar enclaves (en is het aantal enclaves correct).

De derde categorie, de *topologische-verwijzingen-correctheid-constraints*, gaat verder en kijkt niet alleen of een verwijzing bestaat maar ook of deze correct is. Een aantal voorbeelden in deze categorie zijn:
- hebben twee opeenvolgende grenzen ook daadwerkelijk hetzelfde perceel aan dezelfde kant liggen? Feitelijk zijn dit acht controles want er zijn vier winged-edge verwijzingen die elk + of – kunnen zijn;
- komen twee opeenvolgende grenzen daadwerkelijk op hetzelfde punt uit (acht varianten);
- heeft de grens van een eiland inderdaad het ouderperceel aan de juiste kant;
- indien een perceel naar een bepaalde grens wijst, wijst dit perceel dan ook weer terug naar het oorspronkelijke perceel.

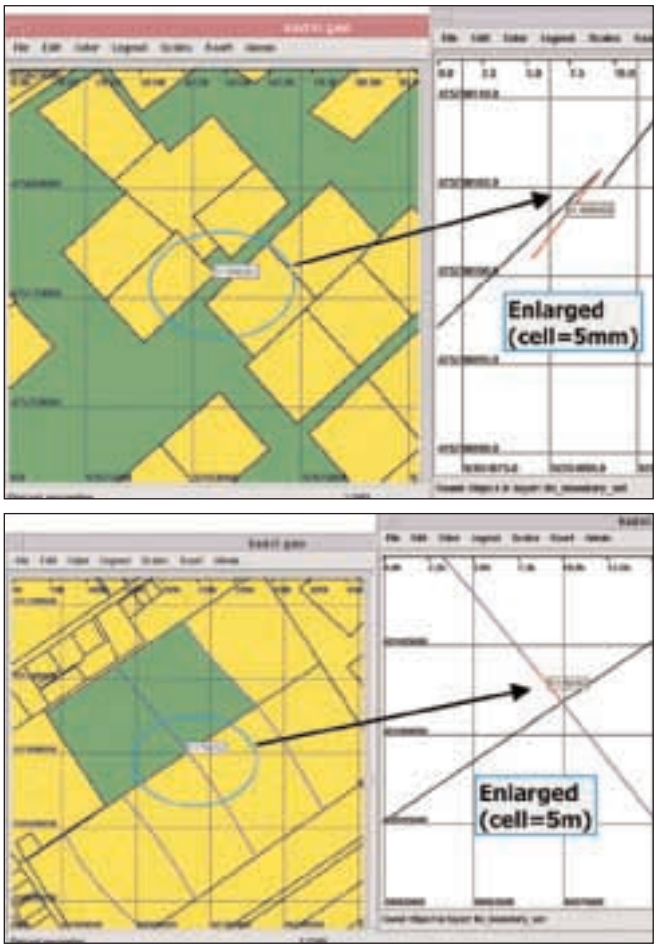

Fig. 4. Enige metrische fouten in de Kadastrale dataset (boven: gaatje tussen twee grenzen, onder: rechte lijn gecodeerd als cirkelboog).

De volgende groep zijn de *referentiële-integriteits-constraints*. Deze zijn in de context van het Kadaster erg belangrijk (heeft elke perceel een eigenaar?). Hiervoor moeten LKI- en AKR-gegevens bij elkaar worden gebracht, bijvoorbeeld in de querytool-omgeving. Een laatste categorie kadastrale constraints bestaat uit *temporele constraints*, zoals twee opeenvolgende versies van een perceel moeten precies aansluiten in de tijd. Door het check-in proces hebben, per definitie, de voorganger en opvolger identieke tijden (maar het zou geen kwaad kunnen dit toch ook af te dwingen via een temporele constraint).

De kadastrale gegevens kunnen als 'clean' worden beschouwd en het systeem is ook ontworpen opdat bepaalde typen fouten (zoals overlappende percelen) helemaal niet kunnen worden gemaakt door het gebruik van een topologische structuur voor percelen. In 1997 zijn bij de conversie van het oude LKI naar het nieuwe LKI alle topologische fouten opgelost. Verder bevat zowel de edit, als de database check-in omgeving ingebouwde controles om topologische fouten te voorkomen. De constraints zijn echter nog niet op databaseniveau geïmplementeerd en bij het draaien van de controles blijkt dat er toch weer een aantal verschillende foutjes zijn ontstaan (fig. 4 en 5). Een belangrijke les hieruit is: 'vertrouw nooit alleen op de edit/check-in omgeving maar neem constraints ook mee in de database' (deze is vooral belangrijk omdat van hieruit meerdere gebruikers deze gevalideerde data kunnen zien). Ook kunnen bepaalde fouten niet schadelijk zijn voor het eigen systeem (zoals 'rechte cirkelbogen' in Ingres) maar wel in een ander systeem (Oracle).

Case 3: bijhouding TOP10NL

Het systeem voor de topografische bijhouding wordt op dit moment geheel vernieuwd en hierbij spelen constraints (volgens de modelgebaseerde aanpak) een belangrijke rol. Naast de vernieuwing van het productiesysteem (van file- naar database-gegevensbeheer) wordt ook het product vernieuwd. Vanaf januari 2006 zal de TOP10NL meer objectgeoriënteerd in plaats van kaartgeoriënteerd zijn, hebben alle objecten een landelijk unieke id en zal het in GML3 geleverd worden.

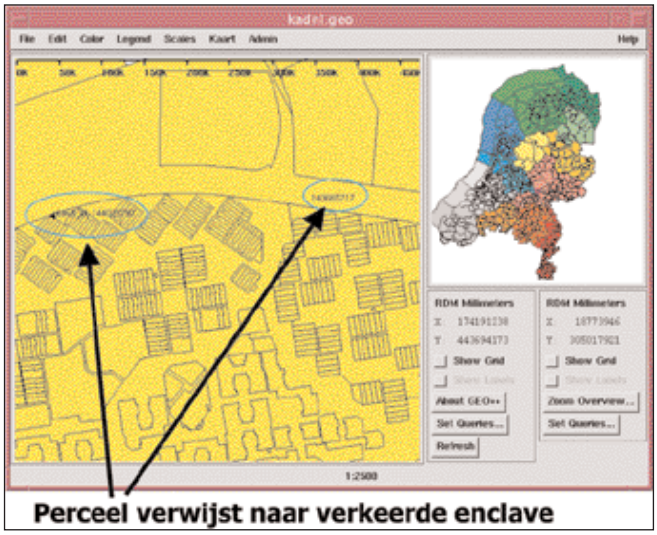

Fig. 5. Enige topologische verwijzingsfouten in de Kadastrale dataset (links: enclave-verwijzing ontbreekt, rechts: perceel verwijst naar verkeerde enclave).

De constraints in de TOP10NL zijn oorspronkelijk ontworpen om ervoor te zorgen dat het dataconversieproces naar het nieuwe productiesysteem schone data oplevert. Dezelfde constraints kunnen echter ook gebruikt worden tijdens het productie-editten in het nieuwe systeem (dit is met succes getest in een prototype) en tijdens de database check-in. De constraints zijn in een bronmodel gedefinieerd, samen met de andere model-elementen (klassen, attributen, relaties). Dit model is gecodeerd in een eigen XML-formaat ontworpen door Vertis en de Topografische Dienst. Hierbij zijn vijf typen constraints onderkend (tabel op deze pagina).

Hier volgt nu een voorbeeld van een constraint van het type topologie. Aangezien het topografische model niet gebaseerd is op een topologische structuur, maar op losse objecten (punt, polyline en polygon features), zien de topologische constraints er nu heel anders uit dan in het – topologisch gestructureerde – kadastrale systeem. De constraints zijn nu nog belangrijker omdat er geen enkele andere functionaliteit is om de topologische kwaliteit van de data te garanderen. Het XML-fragment in het kader toont de constraint die ervoor zorgt dat twee wegen 'WEG_VLAK' niet op zelfde hoogteniveau overlappen (let op gebruik van de ESRI ArcGIS topologie predikaat 'AREAOVERLAPAREA').

Typen constraints in het systeem van de topografische bijhouding met voorbeelden

1. enkele entiteit, enkel attribuut (thematisch)
   0 < weg.aantal_banen <= 10
2. enkele entiteit, meerdere attributen (thematisch)
   weg.A_nummer <> '' dan weg.type='snelweg'
3. geometrie (naast regels zoals minimale lengte van een lijn of oppervalk van een vlak)
   weg.breedte_klasse '<2m' dan geometrie is lijn anders vlak
4. topologie (verschillende subtypes: bedekt_zonder_gaten, geen_overlap, samenvallen,...)
   weg, water and terrein mogen niet overlappen op zelfde hoogte niveau
5. relaties
   elk feature moet ten minste een bron hebben (metadata)

```
<AttribuutRegel>
    <Nummer>top04</Nummer>
    <VervolgNummer/>
    <Categorie/>
    <Beschrijving>Indien Wegvlak overlapt met Wegvlak
        dan moet HOOGTENIVEAU verschillend zijn</Beschrijving>
    <FoutMelding>Wegvlak overlapt Wegvlak</FoutMelding>
    <TriggerNiveau>1</TriggerNiveau>
    <VervolgOperator/>
    <FeatureKlasse>EDT_WEG_VLAK</FeatureKlasse>
    <AlsAttribuut>OBJECTID</AlsAttribuut>
    <AlsOperator>AREAOVERLAPAREA</AlsOperator>
    <AlsWaarde>EDT_WEG_VLAK</AlsWaarde>
    <DanAttribuut>HOOGTENIVEAU</DanAttribuut>
    <DanOperator>!=</DanOperator>
    <DanWaarde>FEATURE2.HOOGTENIVEAU</DanWaarde>
</AttribuutRegel>
```

Het feit dat de constraints nu inderdaad in het bronmodel zijn gespecificeerd en worden gebruikt voor de realisatie van de verschillende subsystemen, is een flinke stap voorwaarts. Dezelfde constraints worden nu zowel bij de conversie als bij het interactief editten (fig. 6) alsook bij de database tijdens check-in gebruikt. Natuurlijk kan het altijd nog beter:

- het model zelf zou in UML gespecificeerd moeten worden, gebaseerd op de OGC/ISO TC211-standaarden;
- in de plaats van een eigen (XML-)codering van constraints, zou het gebruik van een standaard beter zijn (zoals OCL, de Object Constraint Language, zie sectie 'Specificeren van Constraints');
- het uitwisselingsformaat wordt niet automatisch uit hetzelfde bronmodel (zoals voor edit- en databasesubsystemen) afgeleid;
- de constraints zijn nog niet meegenomen in het uitwisselingsformaat: mogelijk zou een deel ook kunnen worden meegenomen in GML/XML-schema's (bijvoorbeeld de domein-constraints).

Fig. 6. Een fout ontdekt tijdens het interactief editen van topografische data.

Case 4: web feature service

Deze case is ook gerelateerd aan het kadastrale toepassingsdomein maar de context is compleet anders: een internet-GIS-toepassing. Hier spelen weer andere aspecten bij implementaties van constraints. Met de beschikbaarheid van de OGC Web Feature Service (WFS) [OGC, 2002] is het voor het eerst mogelijk in een internet-omgeving om niet alleen geo-informatie te bekijken maar ook updates uit te voeren, waarbij server en client zich in een gedistribueerde heterogene omgeving bevinden. De evaluatie van WFS-transacties was onderwerp van het afstudeerwerk van Brentjens [Brentjens, 2004] die dit aan de hand van de case 'notaris schets perceelsgrens' heeft onderzocht (fig. 7). Dit onderzoek leidde ook tot een aantal aanbevelingen voor constraints. Het valideren van features in een WFS-context is lastig: de server kan dergelijk controles uitvoeren nadat de client (editor) features heeft veranderd. De client weet echter niet welke constraints er zijn (behalve eenvoudige domein-constraints die kunnen worden opgenomen in XML-schema) en kan hier met editten dus geen rekening mee houden (met alle gevolgen van dien: onverwachte foutmeldingen die terugkomen). Het is nog onduidelijk hoe constraints in zijn algemeenheid via een XML-schema naar een WFS-client kunnen worden doorgegeven. Een andere interessante vraag is of het mogelijk is om datamodel-constraints te vertalen naar de processtappenstructuur van valide transacties. Bijvoorbeeld: een splitsing van een perceel bestaat altijd uit het verwijderen van een oud perceel, invoegen van een nieuwe grens en toevoegen van twee perceelreferentiepunten (in topologische structuur). Hiervoor is nog meer onderzoek nodig.

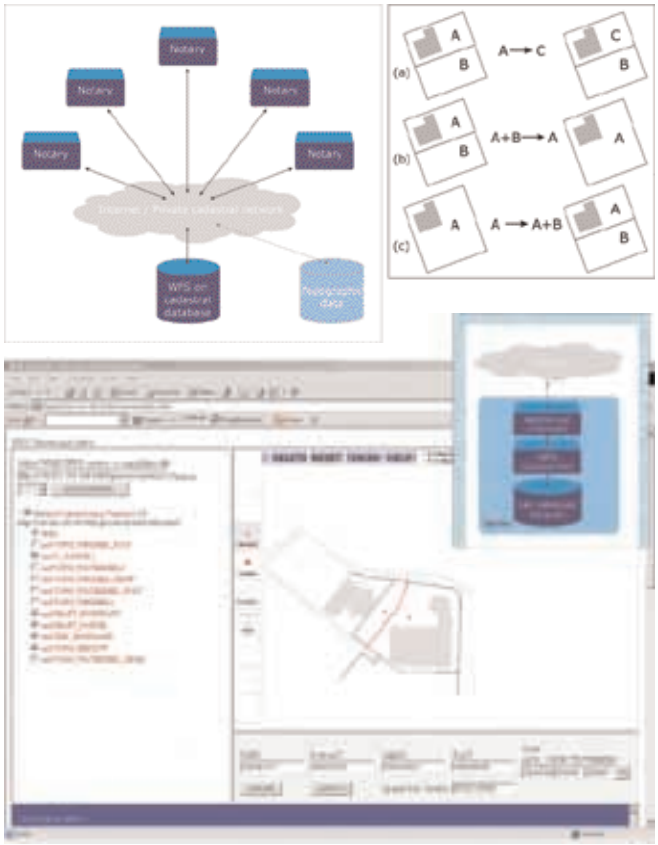

Fig. 7. Case 'Notaris schets perceelgrens via internet' (linksboven: architectuur van de WFS, rechtsboven: een aantal kadastrale edit-operaties, onder: de edit internet WFS-client).

Classificatie van constraints

Om een beter inzicht in constraints en hun gebruik te krijgen, is het belangrijk om ze te classificeren, oftewel: een inzichtelijke taxonomie op te bouwen. Cockroft [Cockcroft, 1997] presenteert een tweedimensionale taxonomie van ruimtelijke constraints: de eerste as betreft statisch tegenover dynamisch en de tweede as kent een onderverdeling in topologische (zoals bij kadastrale en topografische toepassing), semantische (hierbij is ook de betekenis van de objecten van belang) en gebruikersconstraints (overige constraints, vergelijk met business regels in traditionele DBMSen). De tweede as van de indeling van Cockcroft kan worden verbeterd door het onderkennen van vijf sub-assen die kunnen worden gebruikt bij het classificeren van constraints [Louwsma et al, 2005]:

1. het *aantal betrokken objecten* klassen/instanties: één instantie, twee instanties van dezelfde klassen, meerdere instanties van dezelfde klasse, twee instanties van verschillende klassen, of meerdere instanties van verschillende klassen;
2. het *soort attributen en aard van de relaties* tussen de betrokken objecten: metrisch (afstand of hoek tussen twee objecten), topologisch, temporeel, thematisch (semantisch) of gemengd;
3. de *dimensie*: 2D, 3D en mogelijk ook tijd (4D);

4. de manier van uitdrukken: 'mag nooit' (struik mag nooit in het water staan) of 'altijd moet' (struik moet altijd op grasveld geplant worden);
5. de aard van de constraint: 'fysiek onmogelijk' (boom kan niet vrij in de lucht zweven) of 'ontwerpdoelstelling' (struik zou ten zuiden van een boom moeten staan).

Specificeren van constraints

Nu het belang van constraints in verschillende cases is aangetoond en nu we ook beschikken over een classificatie van de verschillende soorten constraints, komt de vraag: hoe nemen we constraints mee in het systeem? Ten eerste zal de specificatie goed toegankelijk en intuïtief moeten zijn voor de makers van de modellen. De constraints zouden onderdeel van het complete model moeten zijn en zo formeel mogelijk moeten zijn zodat hier later automatisch delen van subsystemen (edit/simulatie, database/opslag, uitwisseling/conversie) uit kunnen worden gegenereerd. Een formele aanpak zorgt er ook voor dat de constraints niet ambigue zijn. De Unified Modelling Language (UML) is thans min of meer de standaard aanpak voor het object-georiënteerd modelleren [OMG, 2002]. UML is een grafische taal die verschillende soorten diagrammen onderscheidt waarvan in onze context het klassendiagram het meest relevant is. Aangevuld met een niet-grafische taal binnen UML, de Object Constraint Language (OCL), kunnen constraints verder formeel worden gespecificeerd.

Implementatie van constraints

De constraints zijn gespecificeerd en worden beheerd in een 'repository' maar zouden nu ook het fundament voor verdere implementatie moeten vormen in alle subsystemen: edit/simulatie, database/opslag, uitwisselen/conversie. De edit/simulatie-applicatieomgeving (hoewel belangrijk voor interactie) zal hier niet verder worden uitgewerkt. In de context van de uitwisseling speelt XML een belangrijke rol en kan het XML-schema worden gebruikt voor het beschrijven van het model. Zoals eerder aangegeven is ook hier nog nader onderzoek nodig om de constraints automatisch van UML/OCL om te zetten naar XML-schema.

Nu zal verder worden ingegaan op de omzetting van UML/OCL-constraints naar de implementatie hiervan in een database. De redenen hiervoor zijn dat de database naar alle verwachting hier geschikt voor is (flexibiliteit biedt bij specificeren van constraints) én dat de database de plek is waar vanuit meerdere gebruikers toegang hebben tot dezelfde set consistente gegevens. Sinds SQL92 zijn 'general constraints' (assertions) onderdeel van deze standaard. Helaas zijn ze niet geïmplementeerd in de gangbare database. De ontwikkelaars worden verwezen naar het gebruik van 'triggers' (en vaak ook 'procedures') voor de implementatie van generieke constraints. De soorten constraints die op dit moment wel goed door database worden ondersteund zijn: domein-constraints (specificeren van enumeratie-typen of range-typen) en 'base table constraints' (waaronder ook de referentiële integriteit-constraints vallen). Assertions vormen echter een handige en compacte SQL-schrijfwijze van constraints en het zou ook mogelijk moeten om zijn deze automatisch uit UML/OCL af te leiden. Ze vormen dan mooie opstapjes die kunnen worden gebruikt bij het later herschrijven naar de trigger/procedure vorm.

Het voordeel is dat de trigger/procedure-oplossing echt werkt in de praktijk maar het nadeel is dat deze veel codeerwerk oplevert. Er zijn echter ontwikkelomgevingen die de generatie van deze triggers (en procedures) ondersteunen, zoals Oracle's CDM Ruleframe. Een laatste opmerking over de ondersteuning van constraints door databases: naast standaard domein-constraints en referentiële integriteit komt er de laatste tijd ook steeds meer standaard ondersteuning voor topologische structuren en/of constraints in de databases: Oracle 10g spatial omvat ondersteuning voor topologische structuren (DBMS controleert topologische consistentie), LaserScan Radius topology en er komen ook meer 'middleware'-achtige oplossingen met topologische constraints ondersteuning zoals de ESRI geodatabase.

Conclusie

Hoewel constraints binnen GIS tot nu toe niet veel aandacht kregen, blijken ze in de vier cases een belangrijke rol te spelen, steeds op een andere manier. Er is op basis van de cases en de beschikbare literatuur een voorstel voor de classificatie van constraints gegeven. Deze kunnen dan meegenomen worden in de formele beschrijving van het model (UML/OCL). Uit dit model moeten dan automatisch de verschillende subsystemen worden gegenereerd (edit, opslag, uitwisseling) die dan allemaal gebaseerd zijn op dezelfde constraints. Op dit moment is het volledig automatisch afleiden van de verschillende subsystemen uit het model nog onderwerp van onderzoek. De visie zal voor een deel voorlopig met de hand moeten worden uitgewerkt. ■

Een uitgebreider, Engelstalig, artikel verschijnt binnenkort in: Oosterom, Peter van, Constraints in Spatial Data Models in a Dynamic Context., Chapter 4 in: Drummond, J., Billen, R., Forrest, D. and João, E. (editors), Dynamic & Mobile GIS: Investigating Change in Space and Time. Taylor & Francis, 2006.

Literatuur

- Brentjens, T.J. (2004), *OpenGIS Web Feature Services for editing cadastral data; Analysis and practical experiences*. MSc Thesis Geodesie, TU Delft.
- Cockcroft, S. (1997), *A Taxonomy of Spatial Data Integrity Constraints,* in GeoInformatica jaargang 1, nr. 4, p. 327-343.
- Date C.J., Hugh Darwen (1997), *A guide to the SQL standard, 4th edition.* Addison-Wesley, p. 197-218.
- Lammeren, R. van, et al (2002), *Virtual Reality in the landscape design process*, in: D. Ogrin, I. Marusic en T. Simanic, *Landscape planning in the era of globalisation*, p. 158-165.
- Louwsma, J. (2004), *Constraints in geo-information models, applied to geo-VR in landschape architecture*. MSc Thesis Geodesie, TU Delft.
- Louwsma, J., Zlatanova, S., Van Lammeren, R. en Van Oosterom, P. (2005), *Constraints in models and implementations of (VR) geo-info systems*. Paper op Auto-Carto 2005, Cartography and Geographic Information Society (CaGIS), 18-23 maart 2005, Las Vegas, VS.
- OGC (2002), Vretanos, P. (ed.), Web *Feature Service Implementation Specification, version 1.0.* Reference number: OGC 02-058, OpenGIS Consortium Inc.
- OMG (2002), *Unified Modeling Language Specification (Action Semantics), UML 1.4 with action semantics*. Object Management Group.

Samenvatting

Modelleren van constraints in GIS

In dit artikel wordt het belang van constraints (geldigheidsvoorwaarden) in GIS aangetoond aan de hand van een viertal verschillende cases: landschapsontwerp, kadastrale bijhouding, topografische bijhouding en web-feature server-toepassingen. Op basis van deze cases en de bekende literatuur wordt een verbeterde classificatie van ruimtelijke constraints gegeven. Vervolgens wordt een visie gepresenteerd gebaseerd op constraints als een onderdeel van het gegevensmodel, gespecificeerd in UML/OCL. Tenslotte wordt aandacht besteed hoe deze constraint in de verschillende subsystemen (edit, opslag, uitwisseling) zouden kunnen worden geïmplementeerd, met nadruk op de database-implementatie.

Trefwoorden
GIS-Technologie, modellering van gegevens, onderzoek

Summary

Modelling constrains in GIS

This article proves the importance of constraints in GIS based on four scenarios: landscape design, cadastral maintenance, topographic maintenance and Web Feature Server (WFS) applications. From these examples and a literature review, an improved classification of spatial constraints is provided. Then a view on constraints is presented as part of a data model, defined in UML/OCL. Finally, the attention is focussed to the possible implementation of constraints in various subsystems (editing, storage, exchange) with emphasis on database implementation.

Keywords
GIS technology, data modelling, research

Résumé

Modélisation de contraintes en GIS

Dans le présent article l'importance de contraintes en SIG est démontrée à l'aide de quatre cas : conception paysagère, mise à jour cadastrale, mise à jour topographique et applications réseaux. Sur base de ces exemples et de la littérature une classification améliorée des contraintes spatiales est proposée. Après quoi une vision basée sur des contraintes est présentée comme élément du modèle de données, spécifié en UML/OCL. Finalement on prêtera attention à la question de l'implémentation des contraintes dans les différents sous-systèmes (édition, stockage, échange), avec un accent sur l'implémentation de base de données.

Mots-clés
Technologie SIG, modélisation de données, recherche