POLSKIE
TOWARZYSTWO
INFORMACJI
PRZESTRZENNEJ

ROCZNIKI GEOMATYKI 2003

Podstawy metodyczne i technologiczne infrastruktur geoinformacyjnych

Janusz Michalak

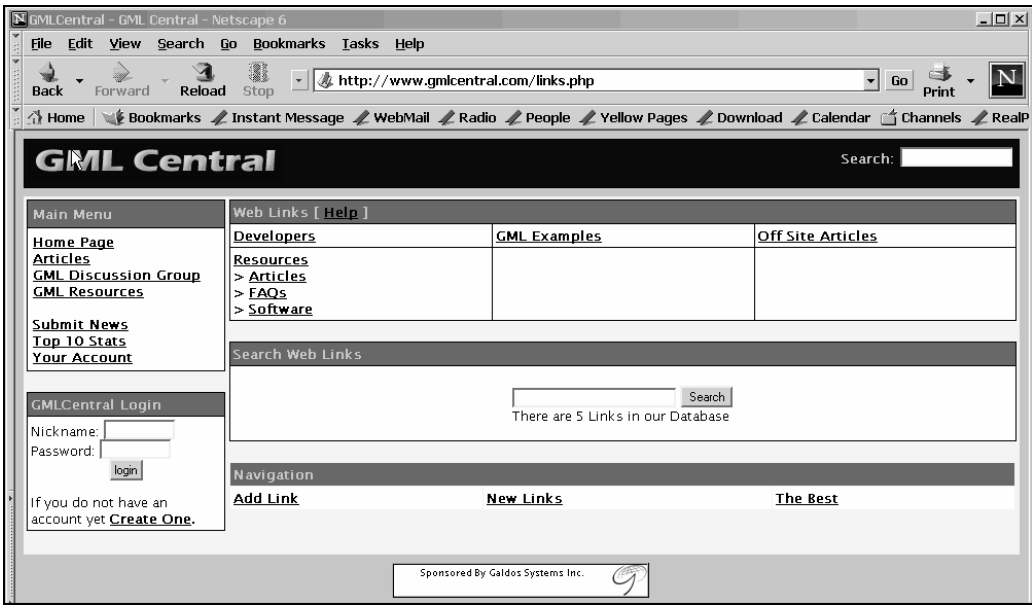
Tom I
Zeszyt 2
Warszawa

JANUSZ MICHALAK
Wydział Geologii Uniwersytetu Warszawskiego
Al. Żwirki i Wigury 93, 02-089 Warszawa
e-mail: J.Michalak@geo.uw.edu.pl
tel. (022) 55-40-529 fax (022) 55-40-001
http://netgis.geo.uw.edu.pl

Spis treści

5. JĘZYK GML *(Geography Markup Language)*

Język GML został opracowany i jest rozwijany przez zespół utworzony przy OGC – obecnie liderami tego zespołu są: Ron Lake (Galdos Systems – Kanada) i Simon Cox (CSIRO – Australia). Jest to aplikacja języka XML dla zapisu geoinformacji. Język GML nie jest w sensie dosłownym językiem aplikacyjnym – nie można go użyć wprost do celów praktycznych. Stanowi on jedynie bazę dla opracowywania języków aplikacyjnych przeznaczonych do określonych zastosowań dotyczących geoinformacji. Szczegółowe opracowania i specyfikacje dotyczące GML można znaleźć w witrynie OGC i w witrynie „GML Central" (rys. 53).



Rys. 53. Witryna internetowa poświęcona aktualnym problematyce języka GML prowadzona przez Galdos Systems. [Źródło: http://www.gmlcentral.org]

5.1. Podstawy języka XML

GML jest aplikacją języka XML i z tego względu reguły posługiwania się tym językiem określa specyfikacja XML i inne specyfikacje dotyczące rozszerzeń XML. Język XML *(eXtensible Markup Language)* jest znacznikowym językiem dla strukturalnego zapisu informacji (Bray, Paoli, Sperberg-McQueen, 1998) – ściślej mówiąc jest metajęzykiem pozwalającym na definiowanie języków specjalistycznych dla różnych obszarów tematycznych. Definicje poszczególnych języków mogą być zapisywane na dwa sposoby:

- starszy sposób posługuje się specjalnym językiem definiowania dokumentów (DTD – *Document Type Definition),*

- nowszy sposób (XML Schema) polega na definiowaniu języków podrzędnych przy pomocy samego języka XML – reguły syntaktyczne i semantyczne są takie same w definicji języka podrzędnego i w zapisach tworzonych w języku podrzędnym opartym na tej definicji.

Chociaż koncepcja języka XML jest stosunkowo nowa – XML w wersji 1.0 został przyjęty w roku 1998 – to pierwszy projekt języka znacznikowego GML *(Generalized Markup Language, prekursora XML)* powstał w roku 1969. Ogniwem pośrednim jest opracowany w roku 1989 i ciągle jeszcze używany język SGML *(Standard Generalized Markup Language)*. Obecnie liczba języków podrzędnych – aplikacji języka XML jest bardzo duża i trudna do określenia – każdy może na własny użytek opracować aplikację XML. Jednak problem polega na tym, aby takie opracowanie zyskało ogólną akceptację środowiska zajmującego się informacją z danej dziedziny – aby to było rozwiązanie standardowe, jak w przypadku języka dla geoinformacji – GML *(Geography Markup Language)*, który jest objęty projektem standardu ISO.

Często można się spotkać z opinią, że język XML jest rozszerzeniem języka HTML i z tego względu jest ograniczony do internetu rozumianego jako WWW. Język XML jest metajęzykiem, a HTML jest aplikacją (językiem podrzędnym) metajęzyka SGML. Odpowiednią nowszą aplikacją języka XML przeznaczoną do redagowania stron WWW jest XHTML. Obie te aplikacje (HTML i XHTML) dotyczą struktury redakcyjnej informacji umieszczanej na stronach WWW, czyli podziału tekstu na elementy strukturalne dotyczące formy: tytuł, podtytuł, paragraf, wyliczenie, definicja, blok tekstu sformatowanego i inne. Pozwala na określenie rozmieszczenia ilustracji i elementów graficznych, a także wielkości i kroju liter, ich koloru i tym podobnych. Według specyfikacji HTML 4.01 (W3C, 1999) atrybuty tekstu mogą być określone przy pomocy trzech sposobów:

- w linii *(inline)* – przy pomocy atrybutu „style”, np.:

Przykład 16.

```
<P style="font-size: 12pt; font-variant: small-caps; color: fuchsia">To jest paragraf</P>
```

- w nagłówku dokumentu, np.:

Przykład 17.

```
<HEAD>
    <STYLE type="text/css">
        P {font-size: 12pt; font-variant: small-caps; color: fuchsia}
    </STYLE>
</HEAD>
```

- lub w oddzielnym pliku, tak zwanym arkuszu stylów (style sheets). Ten sposób pozwala na zmianą wyglądu strony bez modyfikowania dokumentu HTML, na przykład w celu dostosowania wielkości liter do rozdzielczości monitora.

W przeciwieństwie do HTML, w większości zastosowań język XML i jego aplikacje dotyczą struktury informacji pod względem jej treści, czyli aspektu semantycznego tej informacji. Zapis taki może służyć do jej przetwarzania, przechowywania (XML-owe bazy danych), przesyłania (przy pomocy internetu, lecz niekoniecznie WWW) i zobrazowywania – w tym przypadku stosuje się konwersję zapisów strukturalnych pod względem treści do strukturalnej formy – do HTML, XHTML lub SVG.

Język XML jest językiem bardzo uniwersalnym, co potwierdzają fakty stosowania go z powodzeniem prawie we wszystkich dziedzinach. Przyczyną jego uniwersalności jest między innymi jego elastyczność wynikająca z prostoty reguł konstruowania złożonych struktur z prostych fragmentów. Składnia (syntaktyka) tego języka zawiera miedzy innymi następujące pojęcia: element, argument elementu, zawartość elementu, elementy proste i złożone, typ elementu, ograniczenia elementów i ich argumentów oraz przestrzenie nazw. Przestrzenie nazw są związane z semantyką dziedzin zastosowań i pozwalają na bezkonfliktowe używanie tych samych nazw pochodzących z różnych dziedzin w obrębie jednego zapisu XML.

Istotny wpływ na uniwersalność języka XML mają także jego rozszerzenia:

- **XSLT** *(eXtensible Style Language – Transformation)* – rozszerzalny język stylów będący częścią XSL. Druga część XSL to **XSL-FO** *(Formatting Objects)* – obiekty formatujące. Rozszerzenie to pozwala na konwersję zapisów XML opartych na jednych specyfikacjach do zapisów opartych na innych specyfikacjach - miedzy innymi dla zobrazowania informacji (np. do XHTML lub do SVG).
- Z XSL związane są rozszerzenia **XPath** i **CSS**. XPath pozwala na zapis wzorców i wyrażeń potrzebnych do transformacji zapisu, a CSS *(Cascading Style Sheets)* – kaskadowe arkusze stylów powiązane z dokumentem XSLT pozwalają jeden dokument XML przekształcić na wiele różnych dokumentów HTML stosując różne sposoby formatowania (Kazienko, Gwiazda, 2002). Schemat takiej operacji przedstawia rysunek 83 w rozdz.5.8.
- **XLink** – opisuje ogólny model wiązania dokumentów, bez określania konkretnych elementów odpowiedzialnych za to wiązanie. Funkcje elementu łączącego może pełnić, każdy dowolny element dokumentu. Ważną częścią elementów łączących są lokalizatory:
  - **URI#id** – pobierany jest cały dokument (zapis), a następnie w jego obrębie jest znajdowana część określona przez identyfikator lub
  - **URI|id** – decyzję o sposobie wiązania podejmuje program aplikacyjny.
  - **XPointer** – jest rozszerzeniem XLink i pozwala na precyzyjne i zarazem elastyczne odwoływanie się do elementów zapisu XML – szczególnie w przypadku, gdy dokument ma złożoną hierarchiczną strukturę, a wskazywany element może w niej występować wiele razy i w różnych miejscach (na różnych poziomach hierarchii).

Pomimo, że XML nie jest ograniczony do internetu i WWW, to jednak rola WWW jako uniwersalnej przestrzeni informacyjnej jest tu niewątpliwa. Z tego powodu WWW jest ściśle związany ze strukturalnym zapisem informacji opartym na XML i rozszerza zastosowanie tego języka do interoperacyjnego wyszukiwania i uzyskiwania informacji w całej tej przestrzeni, złożonej z olbrzymiej ilości rozproszonych baz danych. W tych zastosowaniach języka XML poważny problem stanowi jednoznaczne identyfikowanie rozsianych po wielu miejscach obiektów informacyjnych nazywanych tu dokumentami. W ogólnym ujęciu obiektami informacyjnymi mogą być bardzo różne rzeczy, na przykład dzieło sztuki, poemat, wynik pomiaru lub równanie matematyczne. Różnorodność stosowanych protokółów sieciowych sprawia, że sposób dostępu do takiego obiektu może być też bardzo różny i w konsekwencji jednoznaczność odwołania się do obiektu informacyjnego, na przykład poprzez URL *(Universal Resource Locator)* nie jest w pełni zagwarantowana. Z tego względu wprowadzono pojęcie uniwersalnego identyfikatora zasobu – URI *(Universal Resource Identifier)*, którego zadaniem jest określenie jakiegoś dowolnego obiektu informacyjnego z zachowaniem następujących warunków:

- Identyfikator musi określać obiekt informacyjny jednoznacznie, czyli bez względu na środki techniczne lub informatyczne, w każdej sytuacji ma wskazywać na jeden i tyko jeden taki obiekt.

- Identyfikator musi mieć zakres globalny, czyli nie może mieć znaczenia to, gdzie obiekt się znajduje.
- Identyfikator musi być trwały, czyli nie może być w innym czasie przypisany do innego obiektu ani określony obiekt nie może zmienić swojego identyfikatora.
- Identyfikator musi być dostatecznie precyzyjny bez względu na charakter informacji, jaką określa. Na przykład może wskazywać jedno zdanie w jakimś dokumencie tekstowym, pojedynczy dźwięk w zapisie multimedialnym lub jeden punktowy element mapy cyfrowej.

Metodyka opracowywania schematów z zastosowaniem XML Schema

Język XML jest językiem bardzo prostym, lecz duża swoboda w definiowaniu poszczególnych elementów składowych i całej struktury stwarza sytuacje, w których można opracować schemat niepoprawny lub bezużyteczny. Z tego powodu dla wyjaśnienia przedstawionych w tej pracy przykładów podane są poniżej podstawowe reguły syntaktyczne tego języka:

- Podstawowym fragmentem informacji zapisanej w XML jest „element" ograniczony znacznikiem początku i końca:

Przykład 18.

```
<jakiśElement>Wnętrze_jakiegośElementu</jakiśElement>,
      ↑                                      ↑
znacznik początku                    znacznik końca
```

- Znacznik początku może zawierać atrybuty, na przykład nazwę:

Przykład 19.

```
<jakiśElement nazwa="NazwaJakiegośElementu">.
```

- Element złożony zawiera w sobie inne elementy – proste lub również złożone:

Przykład 20.

```
<elementZłożony><elementProsty>"coś"</elementProsty></elementZłożony>
```

- Bardziej czytelny zapis wymaga rozłożenia pionowego i wcięć dla elementów wewnętrznych (zagnieżdżony):

Przykład 21.

```
<elementZłożony>
        <elementProsty>
                "coś"
        </elementProsty>
        <innyElementProsty>
                "coś innego"
        </innyElementProsty>
</elementZłożony>.
```

- Zarówno wnętrze elementu jak i jego atrybuty zawierają informację. Jednak oba sposoby zapisu informacji nie są równoważne – wnętrze zawiera treść informacji, a atrybuty zawierają informację o informacji zawartej wewnątrz elementu, czyli metadane „lokalne".
- Jeżeli element zawiera informację tylko w formie argumentów, to stosuje się skróconą formę zapisu ograniczoną tylko do jednego znacznika:

Przykład 22.

```
<odwołaniaDoInnegoElementu xlink:href="#innyElement"/>
```

- Informacja zawarta we wnętrzu (treść) i w argumentach ma zdefiniowany typ. Jest szereg typów zdefiniowanych wstępnie – łącznie ponad 40, oto niektóre z nich: *string, boolean, ENTITY, ID, IDREF, QName, binary, decimal, float, uriReference, timeDuration* i *recurringDuration*. Na ich podstawie definiowane są własne typy proste, jeżeli jest taka potrzeba.
- Wszystkie elementy, jakie mogą wystąpić w określonym miejscu zapisu z zastosowaniem aplikacji XML, i ograniczenia dotyczące tych elementów są zdefiniowane w zapisie (dokumencie) zdefiniowanym przez XML Schema. Jak już wspomniani w starszych aplikacjach posługiwano się inną specyfikacją – XML DTD *(Document Type Definition)*. Przykład prostego schematu:

Przykład 23.

```
<?xml version="1.0" encoding="UTF-8"?>
    <xs:schema
        xmlns:xs="http://www.w3.org/2001/XMLSchema"
        elementFormDefault="qualified" attributeFormDefault="unqualified">
        <xs:element name="OkresCzasu">
            <xs:annotation>
                <xs:documentation>Prosty przykład schematu</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Nazwa"/>
                    <xs:element name="Początek"/>
                    <xs:element name="Koniec"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
</xs:schema>
```

i odpowiadającego mu zapisu:

Przykład 24.

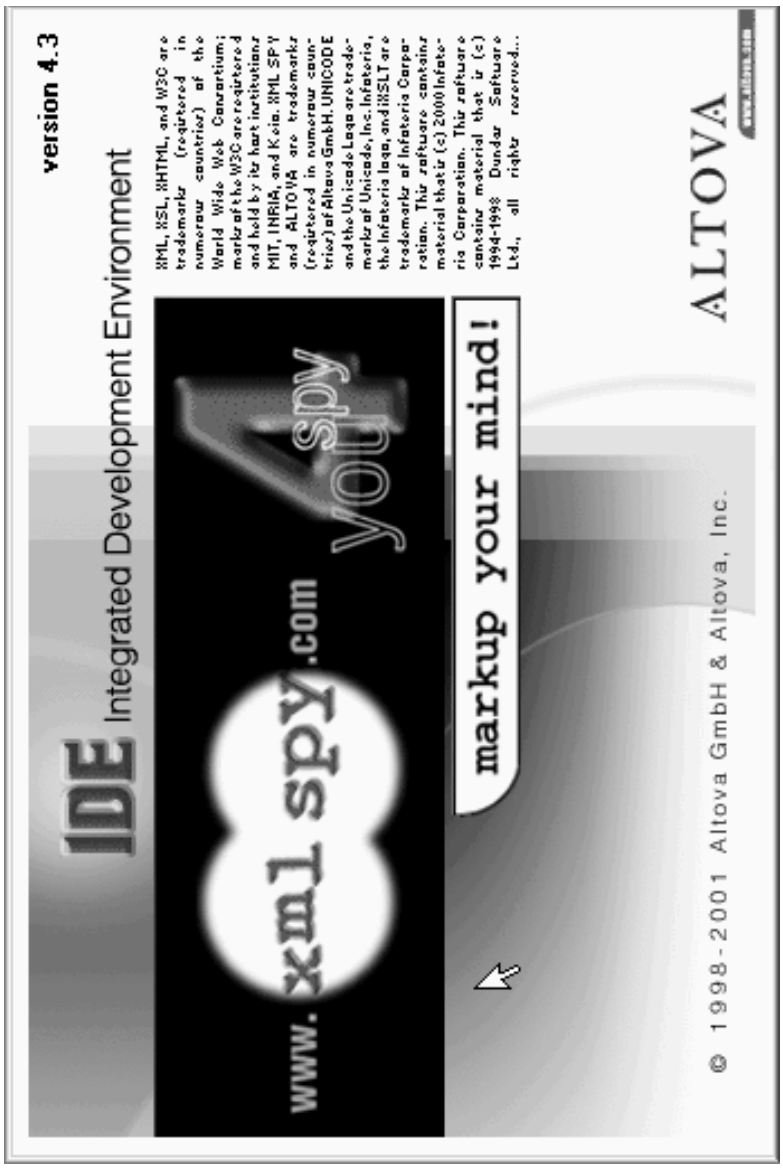
```
<?xml version="1.0" encoding="UTF-8"?>
        <OkresCzasu
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xsi:geol="http://netgis.geo.uw.edu.pl/schemas/czas.xsd">
                <Nazwa>Jakiś okres czasu</Nazwa>
                <Początek>1.1.1996</Początek>
                <Koniec>31.12.2003</Koniec>
        </OkresCzasu>
<!-- ... -->
```

- Stosowane w schematach XML przestrzenie nazw pozwalają na strukturalną organizację wielu schematów ze sobą powiązanych lub od siebie zależnych. Jest to konieczne w przypadkach, gdy jeden zapis opiera się na wielu schematach, na przykład GML3 zawiera 27 schematów posługujących się czterema przestrzeniami nazw.
- Różne języki podrzędne (aplikacje XML) mogą być ze sobą mieszane lub jeden język może być rozwinięciem innego – wykorzystywać schematy drugiego języka. Jednak takie rozwiązania wymagają przestrzegania reguł opracowań standardowych - podziału kompetencji i kompletności w zakresie własnego obszaru tematycznego.

Szczegółowy opis syntaktyki i semantyki języka XML, a także reguł i metod opracowywania schematów i aplikacji jest przedmiotem wielu publikacji (Mercer, 2001; Skonnard, Gudgin, 2001; Stanek, 2001).

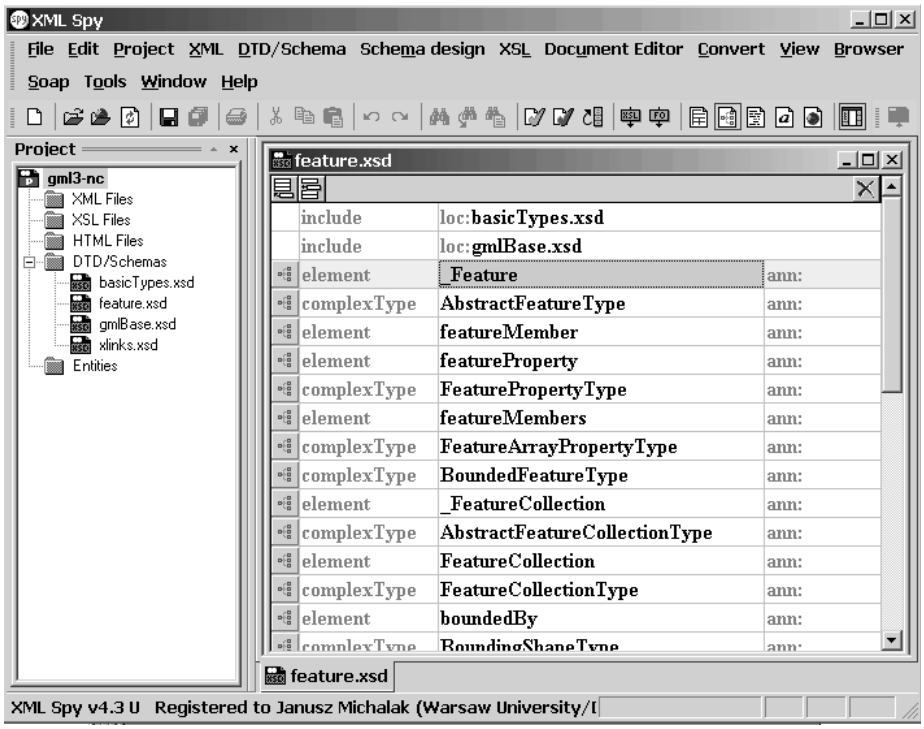
5.2. Oprogramowanie narzędziowe XML Spy

Podobnie jak w przypadku języka UML, opracowywanie dużych i skomplikowanych schematów i dokumentów w języku XML wymaga ścisłej kontroli syntaktycznej poprawności i ich wzajemnej formalnej zgodności. Z tego względu prace takie są wykonywane przy pomocy oprogramowania narzędziowego – edytorów i procesorów. XML-Spy jest najpopularniejszym edytorem i procesorem dla opracowywania i przetwarzania plików XML – w tym także GML. Okno informacyjne tego programu przedstawia rysunek 54.



Rys. 54. Okno informacyjne programu XML-Spy przeznaczonego do opracowywania i edycji dokumentów i schematów XML.

Edytor XML-Spy ma wiele funkcji pozwalających na opracowywanie, weryfikowanie i konwertowanie wszystkich typów dokumentów XML – w tym DTD, XML-Schema, XSL *(eXtensible Stylesheet Language)*. Może on pracować w czterech trybach edycji: zwykłego tekstu, w dwóch trybach tabelarycznych i w trybie diagramów graficznych. Rysunek 55 przedstawia w formie tabelarycznej deklaracje i definicje elementów geoprzestrzennych zawartych w schemacie feature.xsd należącym do specyfikacji GML 3.0.

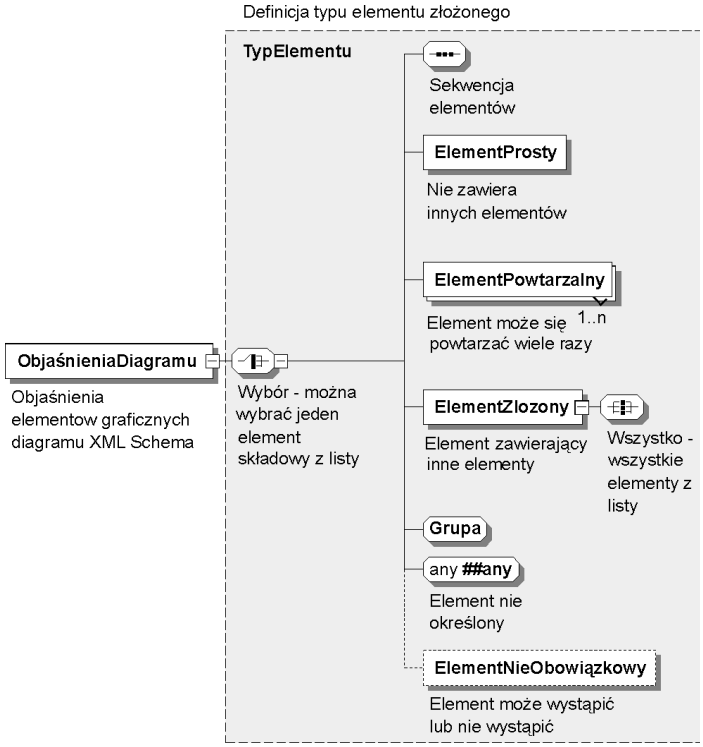


Rys. 55. Edytor XML-Spy w trybie edycji formy tabelarycznej jednego ze schematów języka GML3.

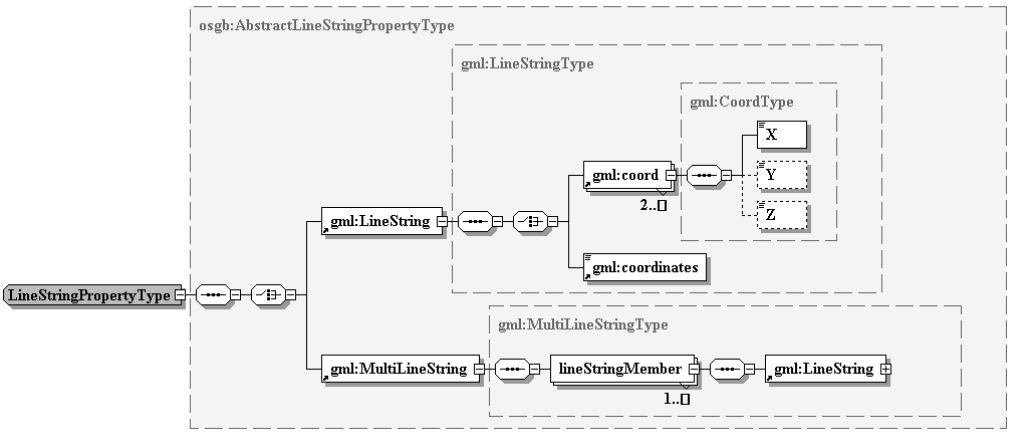
5.2.1. Diagramy XML Spy

Notacja graficzna diagramów jest przedstawiona na rysunku 57 i wyjaśniona na rysunku 56. W przypadku skomplikowanych diagramów jest często niezbędnym narzędziem do zrozumienia i analizy powiązań, jakie występują w takich schematach.

Rysunek 57 przedstawia diagram XML struktury elementu typu „osgb:AbstractLine String-PropertyType". Większość elementów składowych tworzących tą strukturę należy do przestrzeni nazw „gml". Jest to jeden ze sposobów budowania aplikacji w oparciu o GML 3.



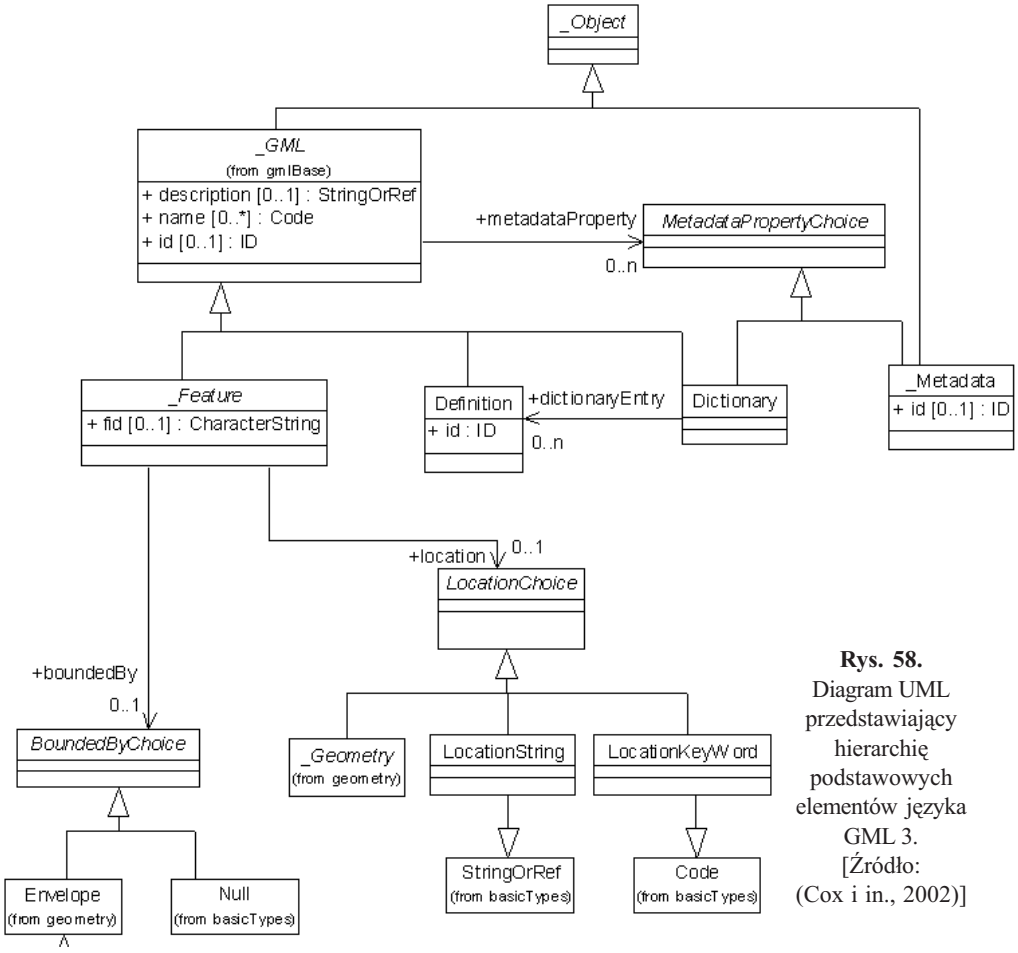
Rys. 56. Objaśnienia diagramów generowanych przez edytor XML_Spy.



Rys. 57. Diagram XML-Spy przedstawiający powiązania elementów zdefiniowanych w deklaracjach typów języka GML i jego aplikacji OSGB (MasterMap).

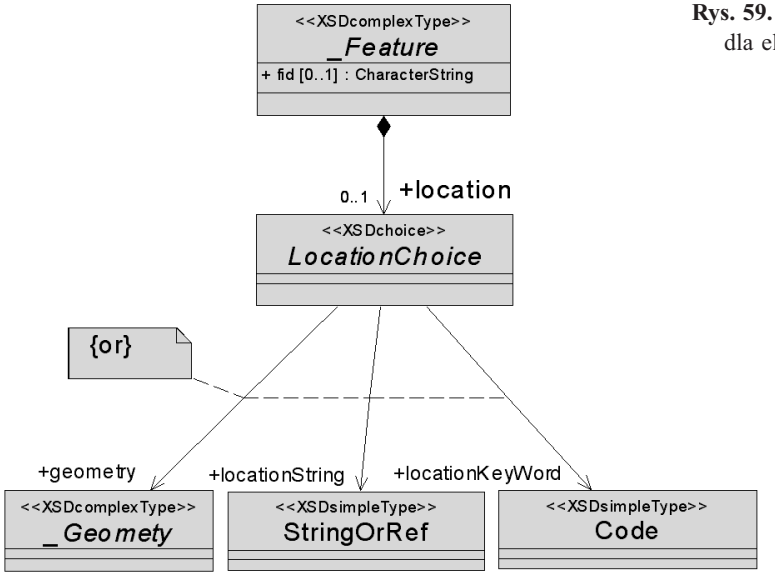
5.3. GML jako aplikacja XML dla geoinformacji

Porównanie trzech kolejnych wersji języka GML, a także jego poprzednika SF-XML, pozwala dostrzec ewolucyjne zmiany w koncepcji i metodyce stosowania XML do geoinformacji. Obecna, najnowsza trzecia wersja różni się znacznie od poprzednich. Różnice te wpływają istotnie na sposób budowania aplikacji tematycznych. Również podstawy ontologiczne i semantyczne tej wersji są znacznie zmienione w porównaniu z poprzednimi. Jest to w zasadzie prawie pełna implementacja podstawowych standardów z grupy ISO 19100. Rysunek 58 przedstawia główny fragment hierarchii klas UML definiujących najważniejsze elementy tej wersji GML.



Rys. 58. Diagram UML przedstawiający hierarchię podstawowych elementów języka GML 3. [Źródło: (Cox i in., 2002)]

Jako pierwsza implementacja tych standardów w XML wersja ta ma wiele fragmentów jeszcze nie całkowicie dopracowanych. Przykładem może być sposób zastosowania konstrukcji „choice", w którym jego elementy podlegające wyborowi są traktowane jako klasy pochodne. Prowadzi to do nieuchronnego podwójnego dziedziczenia, co można zauważyć na przykładzie klasy „LocationString". Bardziej poprawne rozwiązanie tego problemu przedstawia rysunek 59. Taki

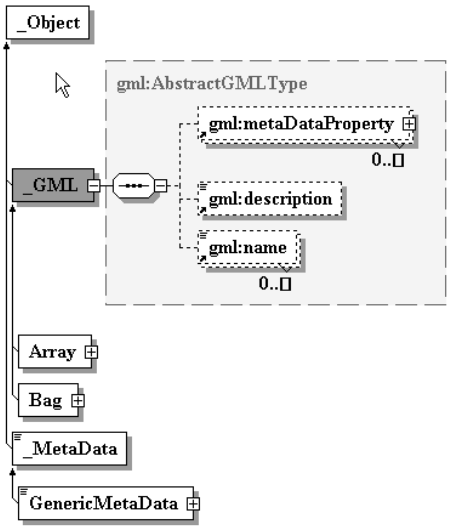


Rys. 59. Poprawiony model UML dla elementów typu Choice.

model elementu „choice" jest zgodny z regułami określonymi w „Profilu UML dla XML" (D. Carlson, 2001)

Również w zakresie metod opracowywania aplikacji ostatnia wersja GML różni się od poprzednich. Autorzy specyfikacji włożyli w tym przypadku wiele trudu, aby reguły opracowywania schematów aplikacyjnych były konsekwentne, precyzyjne i jasno określone. W tym miejscu warto jest zwrócić uwagę na następujące ustalenia i wskazówki:

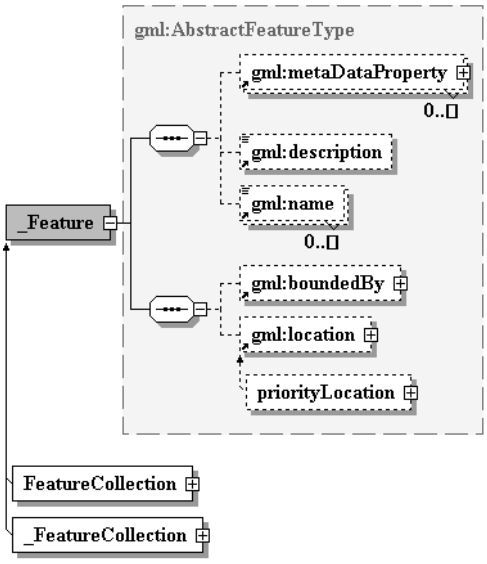
- Najbardziej podstawowym (bazowym) elementem abstrakcyjnym w GLM 3 jest „_Object" (rys. 60). Wszystkie inne elementy są pochodnymi od niego. Zarówno geoprzestrzenne jak i niegeoprzestrzenne.
- Z tego elementu są wyprowadzone dwa inne elementy abstrakcyjne: „_GML" dla różnych rzeczywistych elementów ściśle związanych z informacją geoprzestrzenną i „_MetaData" dla opisu danych.
- Z powyższego powodu w aplikacjach tego języka elementy dotyczące informacji niegeoprzestrzennej powinny być pochodnymi od abstrakcyjnego elementu „_Object".
- Dotyczy to takich elementów jak opis właściciela nieruchomości lub dokumentu określającego własność w systemach ktastralnych.



Rys. 60. Diagram XML przedstawiający główne elementy języka GML 3. Element „_GML" ma wszystkie własne składniki nieobowiązkowe.

- Abstrakcyjny element „_Feature" jest elementem pochodnym od elementu „_Object" – za pośrednictwem elementu „_GML" (Jest to kolejny przykład, że tłumaczenie obu tych terminów na język polski jako obiekt prowadzi do niejednoznaczności i nieporozumień).
- Wszystkie składniki elementu „_Feature" (rys. 61) są nieobowiązkowe i jest on przeznaczony do wyprowadzania tylko rzeczywistych elementów posiadających atrybuty geoprzestrzenne (tylko do wyróżnień).
- Geometria (abstrakcyjny element „_Geometry") nie jest obowiązkowym składnikiem abstrakcyjnego elementu „_Feature".
- „FeatureCollection" jest elementem pochodnym od _Feature.

Bardziej szczegółowy opis reguł opracowywania aplikacji języka GML 3 zawiera rozdział 5.7.



Rys. 61. Diagram XML przedstawiający abstrakcyjny element "_Feature", jego składniki i elementy pochodne.

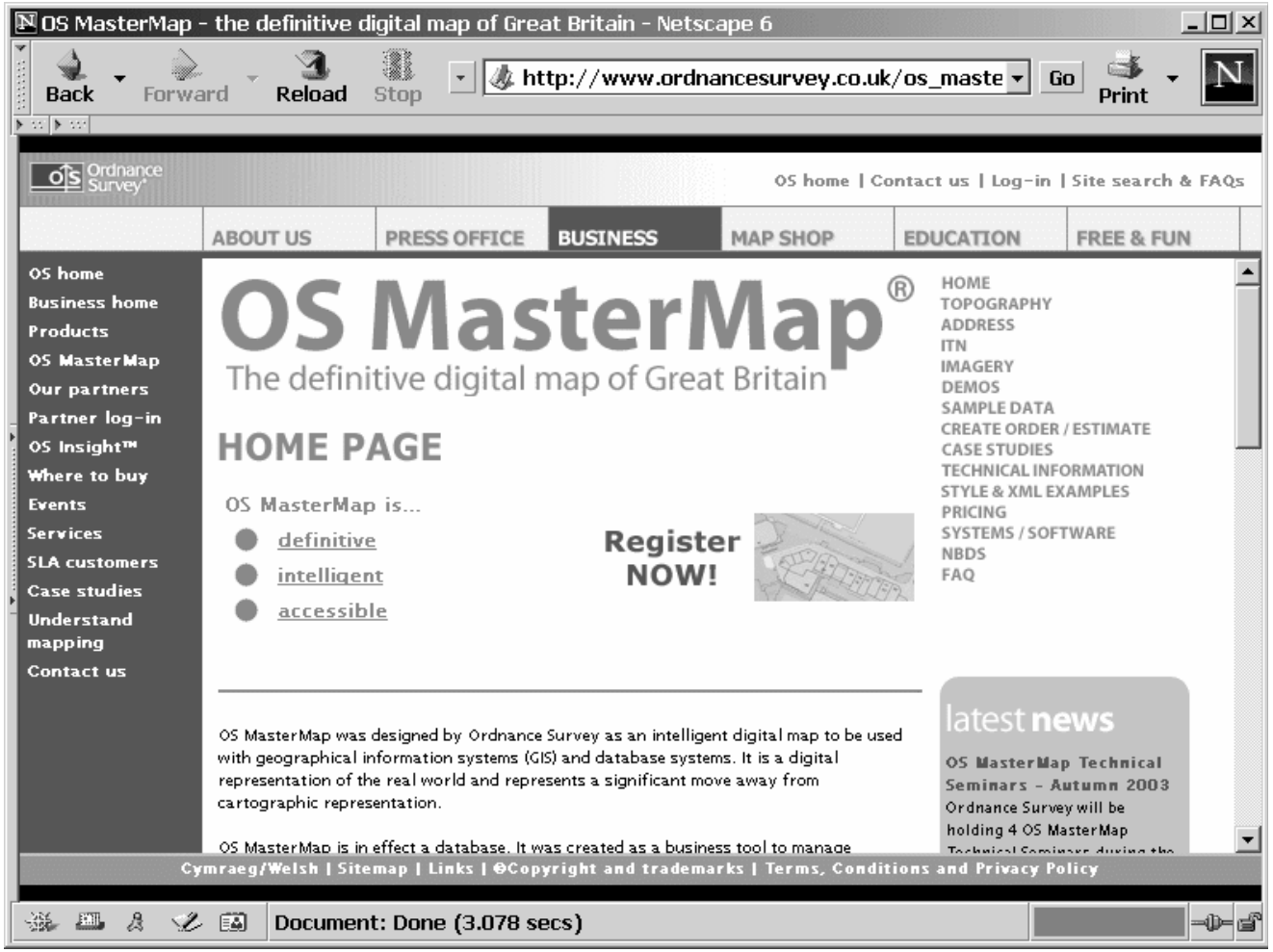
5.4. MasterMap jako przykład zastosowania GML

Brytyjski projekt MasterMap realizowany przez Ordnance Survey jest najbardziej zaawansowaną aplikacją języka GML. MasterMap przechowuje i aktualizuje dane geoprzestrzenne z obszaru Wielkiej Brytanii w skali 1: 1 250. Obejmuje to ponad 417 milionów wyróżnień. Chociaż jest on oparty na starszej wersji GML (2.1), to stanowi dobry przykład praktycznego zastosowania tego języka. Rysunek 62 przedstawia witrynę projektu.

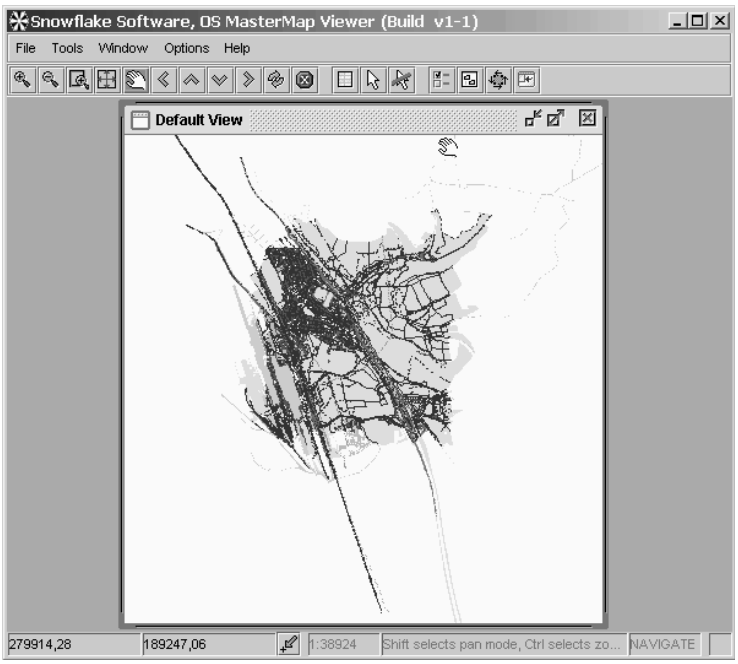
Założeniem tego projektu jest udostępnianie szczegółowych i aktualnych map z całego obszaru Wielkiej Brytanii w formie cyfrowej zapisanej przy pomocy języka GML. Organizacyjne i techniczne założenia projektu przedstawia rozdział 5.4.1. Tu można przyjrzeć się z bliska roli GML, jaką pełni w przyjętej tam technologii i rezultatom, jakie można dzięki niemu uzyskać.

Niezbędnym narzędziem użytkownika jest przeglądarka map GML i w tym przypadku jest to oprogramowanie firmy Snowflake napisane w języku Java. Biorąc pod uwagę powolność aplikacji tego języka wynikającą z konieczności interpretacji kodu pośredniego, przeglądarka ta działa bardzo efektywnie i szybko – nawet w przypadku plików o wielkości 30 megabajtów. Rysunek 63 przedstawia okno tej przeglądarki ze szczegółową mapą miasteczka średniej wielkości. Na rysunku 64 widoczne jest powiększenie tej mapy i pozwala ono ocenić stopień jej szczegółowości.

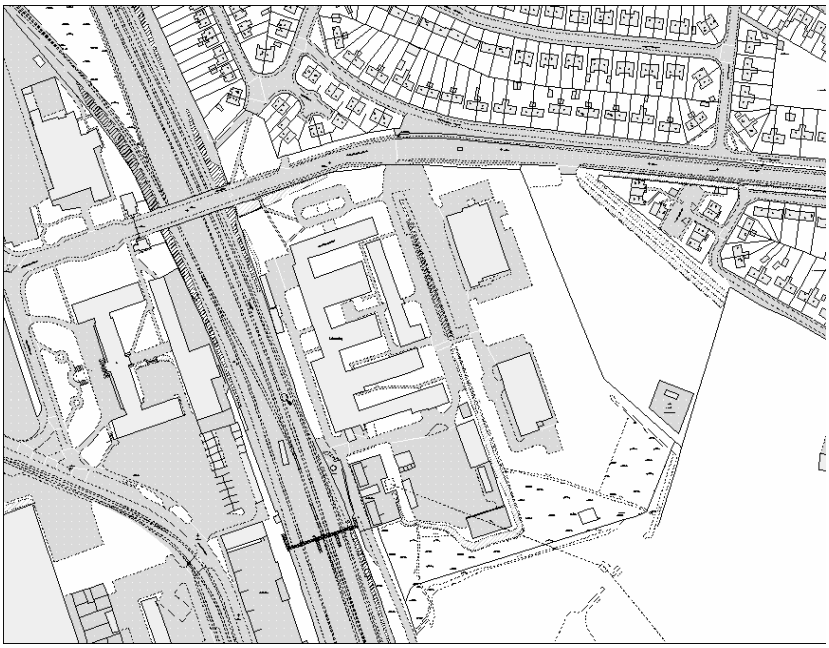
Pomocnicze okna przeglądarki dostarczają informacji niegeoprzestrzennych zapisanych w pliku mapy razem z informacją geoprzestrzenną (rys. 65).



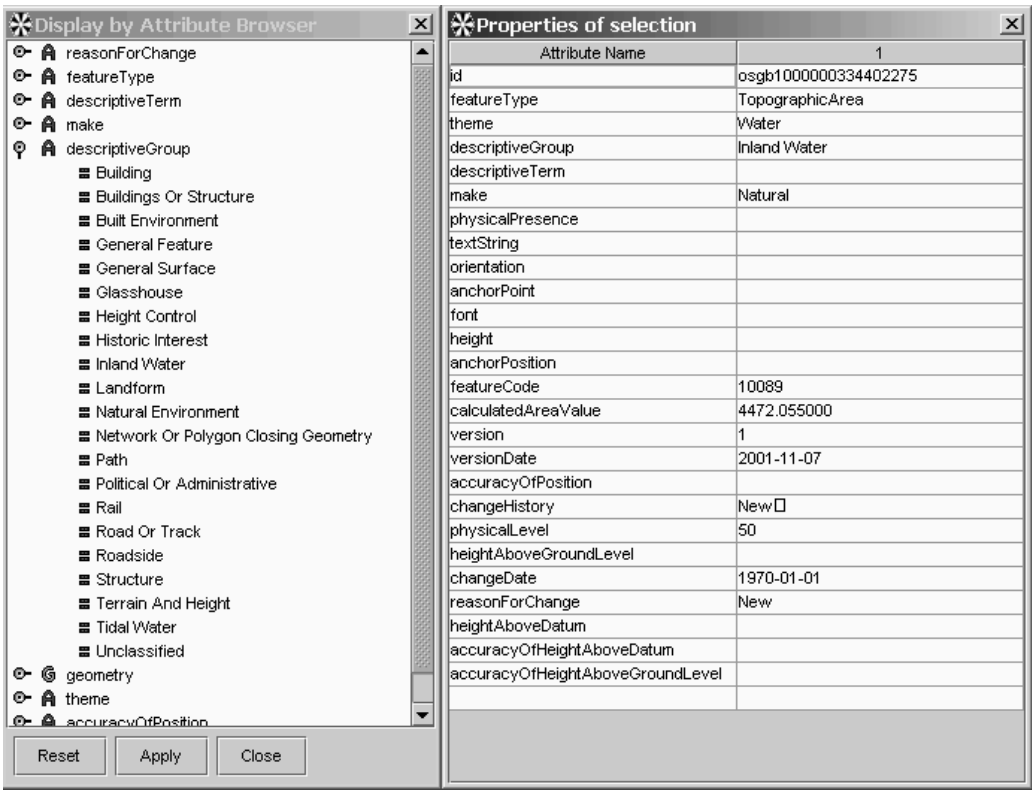
Rys. 62. Witryna internetowa projektu MasterMap realizowanego przez brytyjski Ordnance Survey. [Źródło: http://www.ordnancesurvey.co.uk]



Rys. 63. Przeglądarka map zapisanych w GML firmy Snowflake przeznaczona do aplikacji MasterMap. [Źródło: http://www.snowflakesoft.co.uk]



Rys. 64. Zwiększenie skali mapy pozwala zobaczyć tysiące wyróżnień (wszystkie elementy graficzne i symbole są zapisane wektorowo w XML). [Źródło: http://www.ordnancesurvey.co.uk]



Rys. 65. Pomocnicze okna przeglądarki Snowflake. Okno lewe przedstawia listę opisów wyróżnień (descriptiveGroup). Prawe okno pokazuje tabelę atrybutów niegeoprzestrzennych, jakie posiada wybrane
z mapy wyróżnienie. [Źródło: http://www.ordnancesurvey.co.uk]

Lista opisów wyróżnień widoczna na rysunku 65 (lewe okno) pochodzi ze schematu XML definiującego rozszerzenia aplikacyjne i posługującego się przestrzenią nazw „osgb" w odróżnieniu od przestrzeni nazw „gml" należącej do języka GML. Odpowiedni fragment schematu jest przedstawiony na przykładzie 25. W schematach przestrzeni nazw „osgb" nie stosuje się kodów i słowników, tylko enumeratory zawierające pełne określenia atrybutów wyliczeniowych.

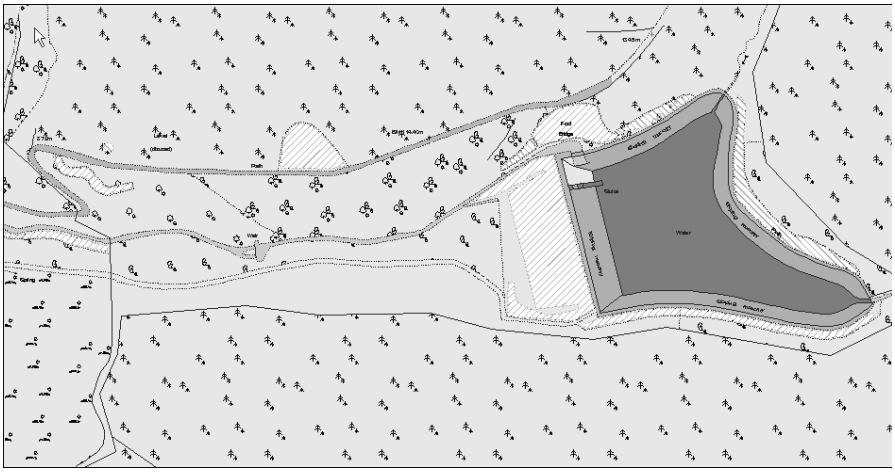
Zobrazowanie geoinformacji zawartej w pliku mapy jest w pełni wektorowe, obok podstawowych elementów geometrycznych (punkt, linia, obszar) również symbole kartograficzne są zapisane wektorowo z zastosowaniem języka XML. Rysunek 66 przedstawia różne typy symboli stosowanych do zobrazowania danych stanowiących treść mapy. Sposób definiowania tych symboli jest przedstawiony w dalszej części tego rozdziału.

Przewiduje się, że w dalszych etapach projektu MasterMap dokumenty GML zawierające dane dla poszczególnych obszarów („mapy" wektorowe) będą uzupełnione ortoobrazami w dużej skali i o wysokiej rozdzielczości. Rysunek 67 przedstawia przykład takiego ortoobrazu, a rysunek 68 jest powiększeniem jego fragmentu.

Aby zrozumieć, w jaki sposób zapis w języku GML zostaje zamieniony na końcowy efekt przedstawiony powyżej, trzeba przeanalizować schematy będące podstawę aplikacji Master-

Przykład 25. [Źródło: http://www.ordnancesurvey.co.uk/]

```
<simpleType name="descriptiveGroupType">
      <restriction base="string">
          <enumeration value="Building"/>
          <enumeration value="Buildings Or Structure"/>
          <enumeration value="Built Environment"/>
          <enumeration value="General Feature"/>
          <enumeration value="General Surface"/>
          <enumeration value="Glasshouse"/>
          <enumeration value="Height Control"/>
          <enumeration value="Historic Interest"/>
          <enumeration value="Inland Water"/>
          <enumeration value="Landform"/>
          <enumeration value="Natural Environment"/>
          <enumeration value="Network Or Polygon Closing Geometry"/>
          <enumeration value="Path"/>
          <enumeration value="Political Or Administrative"/>
          <enumeration value="Rail"/>
          <enumeration value="Road Or Track"/>
          <enumeration value="Roadside"/>
          <enumeration value="Structure"/>
          <enumeration value="Terrain And Height"/>
          <enumeration value="Tidal Water"/>
          <enumeration value="Unclassified"/>
      </restriction>
</simpleType>
```



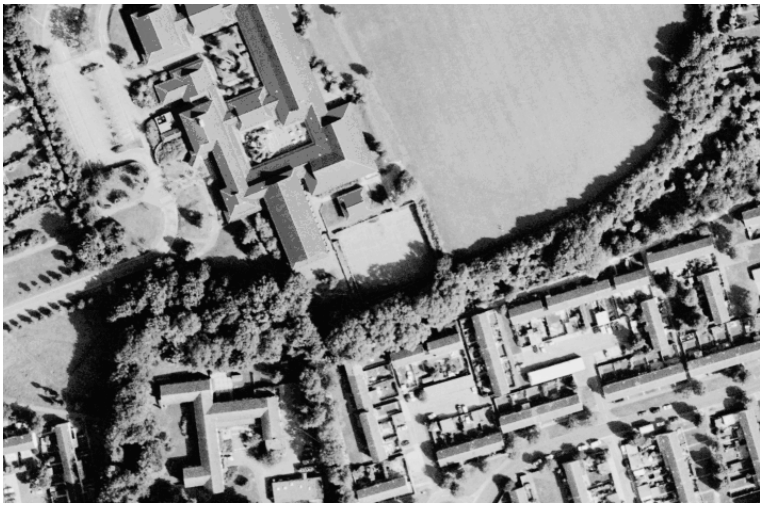
Rys. 66. Fragment mapy jako przykład użycia symboli zapisanych w XML.

Map. Poniższy przykład przedstawia fragment nagłówkowy jednego ze schematów XML projektu MasterMap. Globalne deklaracje elementów zarówno rzeczywistych jak i abstrakcyjnych przestrzeni „osgb" są wyprowadzane z elementów języka GML.

Fragment pliku zapisu mapy MasterMap przedstawiony w przykładzie 27 pokazuje, jak schematy aplikacyjne określają reguły używania elementów języka GML.

Inny fragment (część nagłówkowa) dokumentu MasterMap pokazuje sposób powiązania tego dokumentu ze schematami GML i schematami aplikacyjnymi przestrzeni „osgb". Plik

Rys. 67. Ortoobraz stanowiący uzupełnienie mapy zapisanej w języku GML. [Źródło: http://www.ordnancesurvey.co.uk]



Rys. 68. Powiększenie fragmentu ortoobrazu z rysunku 67 pozwala zobaczyć jego wysoką rozdzielczość. [Źródło: http://www.ordnancesurvey.co.uk]

Przykład 26. [Źródło: http://www.ordnancesurvey.co.uk]

```
<?xml version="1.0" encoding="UTF-8"?>
<schema targetNamespace="http://www.ordnancesurvey.co.uk/xml/namespaces/osgb"
      xmlns:osgb="http://www.ordnancesurvey.co.uk/xml/namespaces/osgb"
      xmlns:gml="http://www.opengis.net/gml"
      xmlns:xlink="http://www.w3.org/1999/xlink"
      xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
      attributeFormDefault="unqualified" version="1.0">
  <annotation>
      <appinfo>OSDNFFeatures.xsd v1.0 2001/08</appinfo>
      <documentation xml:lang="en">Ordnance Survey, (c) Crown Copyright.
          All Rights Reserved August 2001.</documentation>
      <documentation xml:lang="en">See http://www.ordnancesurvey.co.uk for
          guidelines and related information</documentation>
      <documentation xml:lang="en">This schema defines the topographic
          features.</documentation>
  </annotation>
  <!-- include constructs from the OS schemas which import the GML Feature and
          Geometry schemas -->
  <include schemaLocation="OSComplexTypes.xsd"/>
  <!-- ========================================================
  Global element Declarations
  ======================================================== -->
  <!-- feature members-->
  <element name="boundaryMember" type="osgb:boundaryMemberType"
      substitutionGroup="osgb:_featureMember"/>
  <element name="cartographicMember" type="osgb:cartographicMemberType"
      substitutionGroup="osgb:_featureMember"/>
  <element name="topographicMember" type="osgb:topographicMemberType"
      substitutionGroup="osgb:_featureMember"/>
  <!-- Abstract features -->
  <element name="_BoundaryFeature" type="osgb:AbstractFeatureType" abstract="true"
      substitutionGroup="gml:_Feature"/>
  <element name="_CartographicFeature" type="osgb:AbstractFeatureType"
      abstract="true" substitutionGroup="gml:_Feature"/>
  <element name="_TopographicFeature" type="osgb:AbstractFeatureType"
      abstract="true" substitutionGroup="gml:_Feature"/>
  <!-- features -->
```

(przykład 28) zawiera zbiór wyróżnień („osgb:FeatureCollection") ograniczony wielobokiem („gml:Poligon") w formie liniowego pierścienia („gml:LinearRing").

Przykład 29 przedstawia fragment pliku OSDNFFeatures.xsd definiujący typ „punkt topograficzny".

Dla umożliwienia stosowania symboli kartograficznych przy zobrazowaniu danych zawartych w mapie dodatkowe pliki (dokumenty) definiują wektorową postać tych symboli w XML. Przykłady 30 i 31 są zapisem symbolu „drzewo liściaste" i grupy symboli „drzewa liściaste i iglaste". Rysunek 69 przedstawia graficzną postać tych symboli, a rysunek 70 ilustruje zastosowanie ich w obrazie mapy.

Przykład 27. [Źródło: http://www.ordnancesurvey.co.uk]

```
<osgb:cartographicMember>
    <osgb:CartographicSymbol fid="osgb1000000762071457">
        <osgb:featureCode>10130</osgb:featureCode>
        <osgb:version>1</osgb:version>
        <osgb:versionDate>2001-11-07</osgb:versionDate>
        <osgb:theme>Administrative Boundaries</osgb:theme>
        <osgb:changeHistory>
            <osgb:changeDate>1970-01-01</osgb:changeDate>
            <osgb:reasonForChange>New</osgb:reasonForChange>
        </osgb:changeHistory>
        <osgb:descriptiveGroup>Political Or Administrative</osgb:descriptiveGroup>
        <osgb:descriptiveTerm>Boundary Half Mereing</osgb:descriptiveTerm>
        <osgb:orientation>3209</osgb:orientation>
        <osgb:physicalLevel>50</osgb:physicalLevel>
        <osgb:point>
            <gml:Point srsName="osgb:BNG">
                <gml:coordinates>278219.080,187556.600</gml:coordinates>
            </gml:Point>
        </osgb:point>
    </osgb:CartographicSymbol>
</osgb:cartographicMember>
```

Przykład 28. [Źródło: http://www.ordnancesurvey.co.uk]

```
<?xml version="1.0" encoding="UTF-8"?>
<osgb:FeatureCollection
  xmlns:osgb="http://www.ordnancesurvey.co.uk/xml/namespaces/osgb"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2000/10/XMLSchema-instance"
  xsi:schemaLocation="http://www.ordnancesurvey.co.uk/xml/namespaces/osgb
  http://www.ordnancesurvey.co.uk/xml/schema/OSDNFFeatures.xsd" fid="GDS2231">
  <gml:description>Ordnance Survey, (c) Crown Copyright. All rights reserved,
      2001-12-10</gml:description>
  <gml:boundedBy>
      <gml:null>unknown</gml:null>
  </gml:boundedBy>
  <osgb:queryTime>2001-12-10T07:22:45</osgb:queryTime>
  <osgb:queryExtent>
      <gml:Polygon srsName="osgb:BNG">
          <gml:outerBoundaryIs>
              <gml:LinearRing>
                  <gml:coordinates>277971.000,185990.000 280016.000,185990.000
                      280028.000,188025.000 277964.000,188035.000 277971.000,
                      185990.000 </gml:coordinates>
              </gml:LinearRing>
          </gml:outerBoundaryIs>
      </gml:Polygon>
  </osgb:queryExtent>
```

Przykład 29. [Źródło: http://www.ordnancesurvey.co.uk]

```
<complexType name="TopographicPointType">
      <complexContent>
          <extension base="osgb:AbstractFeatureType">
              <sequence>
                  <element name="accuracyOfPosition"
                      type="osgb:accuracyOfPositionType"/>
                  <element name="changeHistory" type="osgb:changeHistoryType"
                      maxOccurs="unbounded"/>
                  <element name="descriptiveGroup" type="osgb:descriptiveGroupType"
                      maxOccurs="unbounded"/>
                  <element name="descriptiveTerm" type="string" minOccurs="0"
                      maxOccurs="unbounded"/>
                  <element name="heightAboveDatum" type="osgb:heightAboveDatumType"
                      minOccurs="0"/>
                  <element name="heightAboveGroundLevel"
                      type="osgb:heightAboveGroundLevelType" minOccurs="0"/>
                  <element name="make" type="osgb:makeType" minOccurs="0"/>
                  <element name="physicalLevel" type="osgb:physicalLevelType"/>
                  <element name="physicalPresence" type="osgb:physicalPresenceType"
                      minOccurs="0"/>
                  <element name="point" type="gml:PointPropertyType"/>
                  <element name="referenceToFeature" type="gml:FeatureAssociationType"
                      minOccurs="0"/>
              </sequence>
          </extension>
      </complexContent>
  </complexType>
```

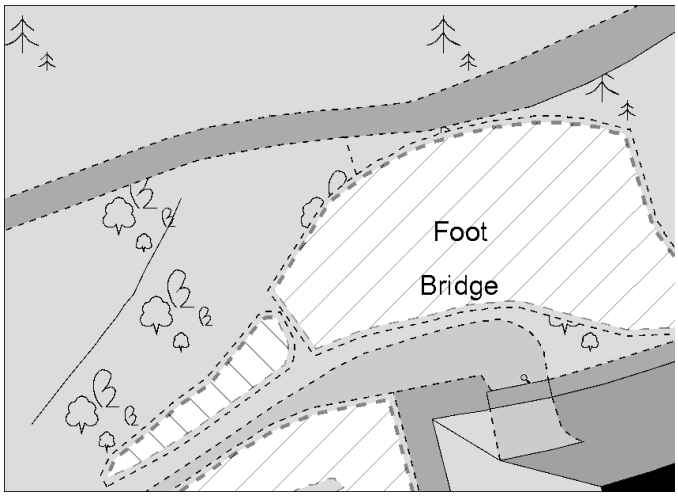
Przykład 30. [Źródło: http://www.ordnancesurvey.co.uk]

```
Geometry: nonconiferousGeometry
Arc Geometry:
<path d="M0,-1.6L-0.2,-0.8a0.6 0.6 0 1 0 -0.8 0.86a0.55 0.55 0 0 0 0.45 0.89a0.56 0.56 0 0 0 1.1 -
0.0a0.55 0.55 0 0 0 0.45 -0.89a0.6 0.6 0 1 0 -0.8 0.86L0,-1.6z"/>
Line Geometry:
<polyline points='-1.074,0.097 -1.210,-0.031 -1.299,-0.194 -1.334,-0.377 1.312, 0.561 -1.233, -0.730
-1.106,-0.866 –0.944,-0.957 -0.761,-0.993 0.576,-0.972 -0.407, -0.894 -0.270,-0.768
-0.269,-0.765 -0.068,-1.539 0.012,-1.539 0.193,-0.756 0.193, 0.756 0.329,-0.887 0.499,-0.969
0.685, 0.993 0.870,-0.959 1.036,-0.869 1.164, 0.731 1.244,-0.561 1.267, 0.374 1.230,-0.189 1.138,-
0.025 0.999,0.102 1.087,0.259 1.119,0.436 1.094,0.615 1.012,0.775
0.883,0.901 0.720,0.979 0.541,1.000 0.541,1.000 0.464,1.184 0.331,1.331 0.156,1.427
-0.040,1.461 -0.236,1.427 0.411,1.331 -0.544,1.184 -0.621,1.000 -0.801,0.978 -0.964,0.900
-1.093,0.773 1.175,0.611 -1.199,0.431 -1.164,0.254 -1.074,0.097'/>
Symbols
Name: positionedNonconiferousTreeSymbol
Style: stroke:#666666;fill:none;stroke-width:0.087
Name: nonconiferousTreeFillSymbol
Style: stroke:#669966;fill:none;stroke-width:0.087
```

Przykład 31. [Źródło: http://www.ordnancesurvey.co.uk]

```
Name: scrubConiferousTreesAndNonconiferousTreesFillSymbol
Symbol 1: scrubFillSymbol
Symbol 2: coniferousTreesFillSymbol
Symbol 3: nonconiferousTreesFillSymbol
```

Rys. 69. Obraz graficzny symboli zdefiniowanych w przykładach 30 i 31. [Źródło: http://www.ordnancesurvey.co.uk]



Rys. 70. Fragment mapy przedstawiający użycie symboli kartograficznych z rysunku 69. [Źródło: http://www.ordnancesurvey.co.uk]

5.4.1. Projekt systemu obsługi MasterMap

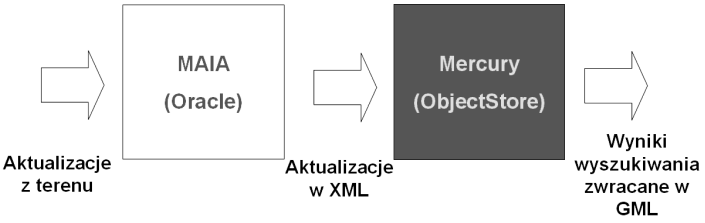
Projekt MasterMap jest największym przedsięwzięciem opartym na języku GML, a technologia zapisu i wizualizacji danych geoprzestrzennych to tylko jeden z aspektów tego złożonego przedsięwzięcia. Oddzielnym zagadnieniem jest zorganizowanie systemu zbierania danych, ich przetwarzania, aktualizacji i udostępniania. Rysunek 71 przedstawia ogólny schemat przepływu danych korzystający z dwóch systemów:

- Pierwszym jest obiektowo-relacyjna baza danych MAIA zarządzana przez Oracle. Do bazy tej są wprowadzane dane i ich bieżące aktualizacje z terenu.
- Drugi system (Mercury), stanowiący podstawowy zasób danych geporzestrzennych, jest oparty na w pełni obiektowej bazie zarządzanej przez system ObjectStore.

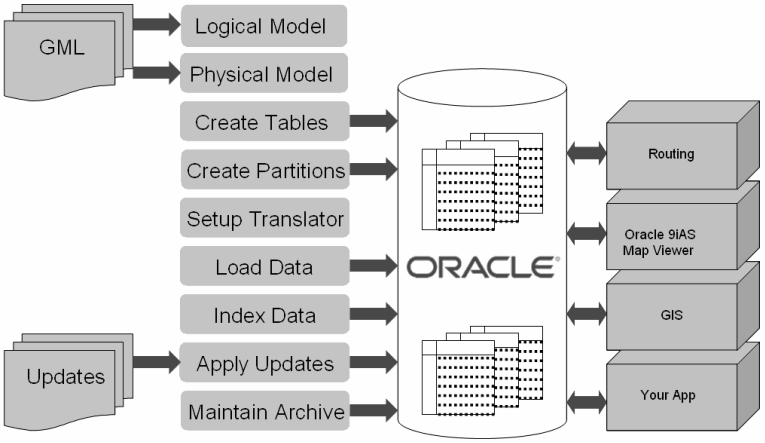
Przepływ danych pomiędzy tymi systemami dzieli się na trzy etapy:

- MAIA zbiera i przechowuje aktualizacje danych z pomiarów w terenie.
- Aktualizacje są przesyłane w języku XML do systemu Mercury i przechowywane tam jako obiekty C++ (pod OSZBD ObjectStore).
- Mercury udostępnia dane na żądanie klientów w języku GML.

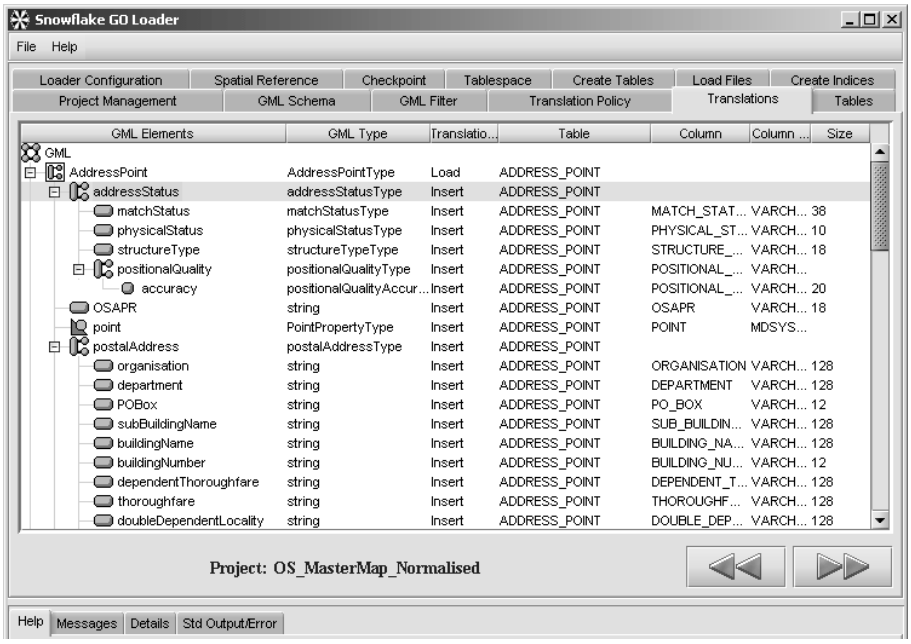
Obecnie system obsługi MasterMap jest modyfikowany i rozbudowywany. Zakłada się, że wszystkie dane przepływające do i od tych systemów będą zapisane w języku GML. Istotny element całego systemy będzie stanowił GO-Loader – oprogramowanie firmy Snowflake przeznaczone do konwersji danych MasterMap (rys. 72). Pozwala ono dane zapisane w GML (zbiory główne i aktualizacje) „załadować" do bazy zarządzanej przez Oracle (rys. 73)



Rys. 71. Główne elementy złożonego i rozproszonego systemu obsługi MasterMap. [Źródło: http://www.ordnancesurvey.co.uk]



Rys. 72. Schemat przedstawiający funkcje realizowane przez GO-Loader firmy Snowflake [Źródło: http://www.snowflakesoft.co.uk]

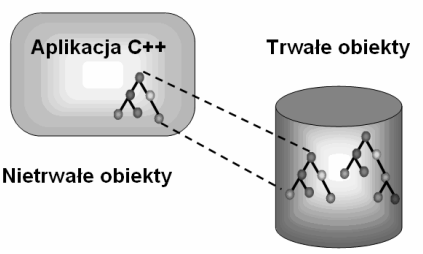


Rys. 73. Okno sterujące konwersja danych MasterMap w programie GO-Loader. [Źródło: http://www.snowflakesoft.co.uk]

Istotnym nowym składnikiem systemu programowego obsługi MasterMap jest baza Mercury zarządzana przez ObjectStore. To rozwiązanie zasługuje na bliższe poznanie ze względu na stopniowe przechodzenie w technologiach geoinformacyjnych od baz obiektowo-relacyjnych do baz „czysto-obiektowych".

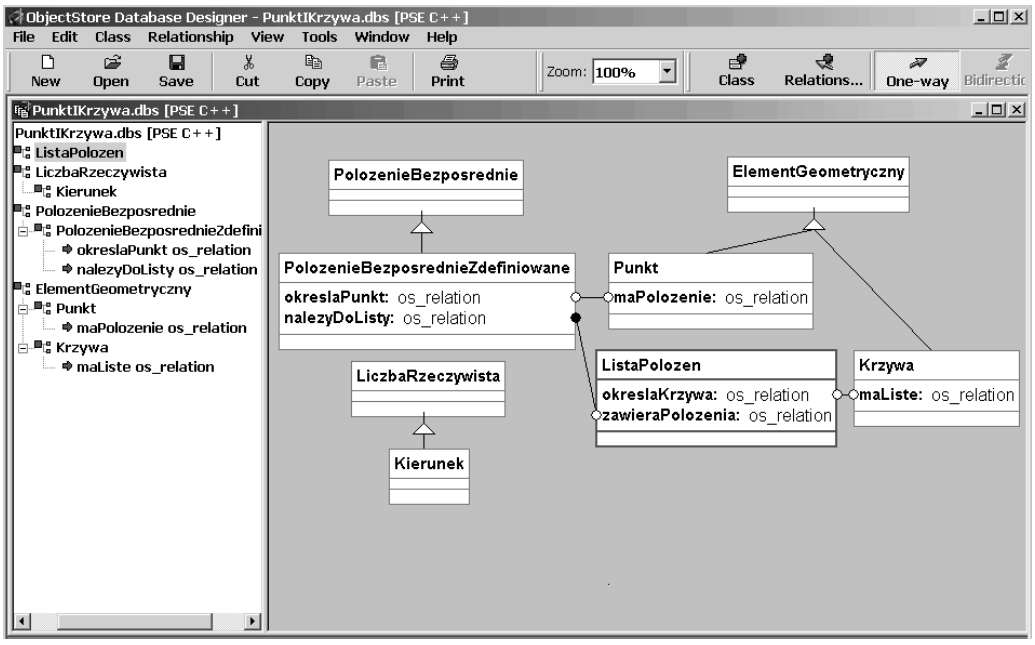
ObjectStore jest „czysto-obiektową" bazą danych (a ściślej: OSZBD – obiektowym systemem zarządzania bazą danych):

- daje ona możliwość trwałego składowania obiektów C++,
- obiekty są składowane w takiej samej postaci jak nietrwałe obiekty C++,
- obiekty nietrwałe są przechowywane na stosie lub stercie (w RAM) i przestają istnieć po zakończeniu działania aplikacji C++,
- obiekty trwałe (w bazie danych) są dostępne i mogą być używane w taki sam sposób jak zwykłe obiekty C++ (rys. 74).



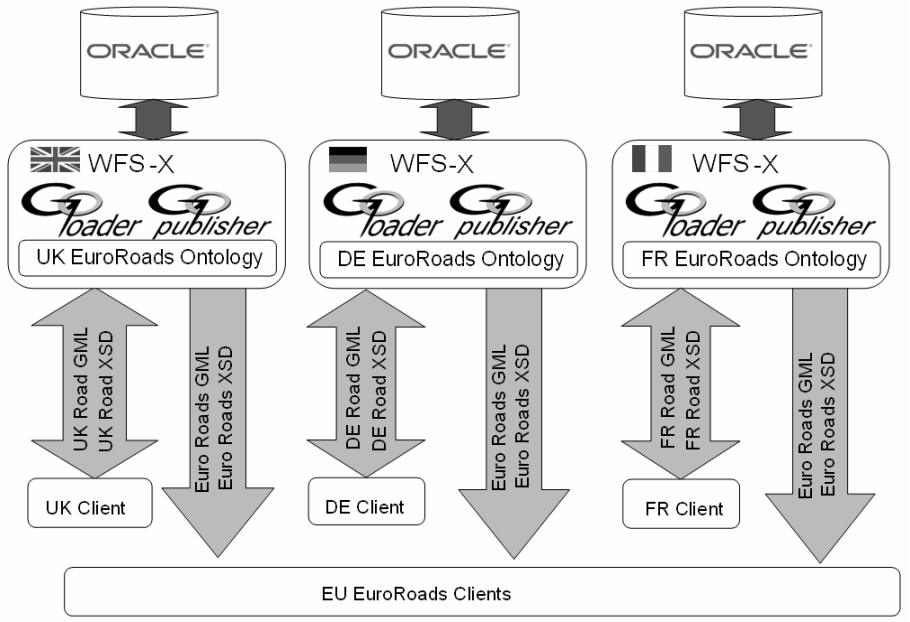
Rys. 74. Schemat mapowania obiektów nietrwałych aplikacji w języku C++ do obiektów trwałych złożonych w obiektowej bazie danych zarządzanej przez ObjectStore. [Źródło: http://www.ordnancesurvey.co.uk]

Projektowanie struktury obiektowej bazy danych w systemie ObjectStore odbywa się przy pomocy programu Designer w języku UML. Przykład przedstawiony na rysunku 75 dotyczy próby zastosowania tego systemu do polskich potrzeb (wynikających z Polskiej Normy) w odniesieniu do punktu i krzywej.



Rys. 75. Okno programu Designer systemu ObjectStore przedstawiające strukturę bazy zapisaną w języku UML.

Oprogramowanie GO-Loader może także stanowić podstawę dla innych zastosowań z zakresu systemów obsługi infrastruktury geoinformacyjnej. Rozszerzenie standardu OGC dla WFS o funkcje ontologiczne i semantyczne pozwala na semantyczną translację geoinformacji pomiędzy różnymi NSDI. Koncepcja ta określona akronimem WFS-X jest tu realizowana przez współpracujące ze sobą systemy GO-Loader i GO-Publisher firmy Snowflake (rys. 76).



Rys. 76. Schemat projektu europejskiej infrastruktury geoinformacyjnej dotyczącej dróg. Projekt ten jest oparty na systemach programowych firmy Snowflake – GO-Loader i GO-Publisher współpracujących z bazami zarządzanymi przez system Oracle. [Źródło: http://www.snowflakesoft.co.uk]

5.5. Deegree GML Viewer/Converter

Oprogramowanie Deegree (objęte licencją GNU – OpenSource) przedstawione w rozdziale 4.6.3 i opisane w rozdziale 6.3.2 jest kolejnym interesującym przykładem zastosowania języka GML. Jeden z jego składników – GML Viewer/Converter ma funkcje umożliwiające konwersje danych zapisanych w różnych formatach do dokumentów XML zgodnych ze schematami GML wersji 2.1. Hierarchę klas Java interfejsu obsługi języka GML przedstawia przykład:

Przykład 32. [Źródło: dokumentacja pakietu Deegree]

```
Interface Hierarchy:
        interface org.deegree.gml.GMLBox
        interface org.deegree.gml.GMLCoord
        interface org.deegree.gml.GMLCoordinates
        interface org.deegree.gml.GMLDocument
        interface org.deegree.gml.GMLFeature
                interface org.deegree.gml.GMLFeatureCollection
        interface org.deegree.gml.GMLGeometry
                interface org.deegree.gml.GMLGeometryCollection
                        interface org.deegree.gml.GMLMultiLineString
                        interface org.deegree.gml.GMLMultiPoint
                        interface org.deegree.gml.GMLMultiPolygon
                interface org.deegree.gml.GMLLinearRing
                interface org.deegree.gml.GMLLineString
                interface org.deegree.gml.GMLPoint
                interface org.deegree.gml.GMLPolygon
        interface org.deegree.gml.GMLNameSpace
        interface org.deegree.gml.GMLProperty
                interface org.deegree.gml.GMLComplexProperty
                interface org.deegree.gml.GMLFeatureProperty
                interface org.deegree.gml.GMLGeoProperty
        interface org.deegree.gml.GMLSchema
```

5.6. Lista oprogramowania implementującego GML

Tak jak inne aplikacje XML, język GML bez odpowiedniego wsparcia w postaci oprogramowania jest w zasadzie bezużyteczny i może stanowić jedynie przedmiot teoretycznych dyskusji. Z tego względu wiele ośrodków komercyjnych i akademickich prowadzi prace nad oprogramowaniem mogącym generować, przetwarzać i analizować dokumentu zapisane w tym języku. OGC, tak jak dla innych specyfikacji implementacyjnych, również dla GML prowadzi program testowania zgodności i publikuje listę oprogramowania spełniającego wymagania specyfikacji GML. Lista ta jest ciągle uaktualniana i z tego powodu przedstawiony tu poniżej spis (tabela 3) jest jedynie obrazem stanu z dnia 28.08.03.

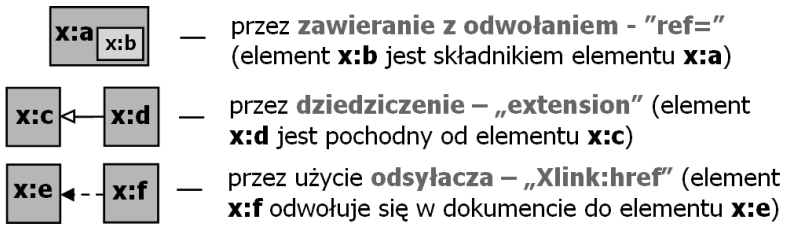
5.7. Reguły opracowywania aplikacji GML

We wstępie do rozdziału 5 jest poruszony problem aplikacji tematycznych (dziedzinowych) tego języka. Tak jak bez odpowiedniego oprogramowania GML jest praktycznie bezużyteczny, również bez rozszerzeń dotyczących informacji tematycznej jego zakres zastosowań jest bardzo ograniczony. Z tego względu zastosowania tego języka wymagają łączenia schematów GML (z przestrzenią nazw „gml") ze schematami dziedzinowymi, nazywanymi tu dla uproszczenia „non-GML" (z przestrzenią nazw „ng").

Rozdział 5.3 zawiera ogólne wskazówki dotyczące opracowywania aplikacji dla nowej wersji 3. Tu problem ten jest rozszerzony o zagadnienie wiązania ze sobą dwóch oddzielnych hierarchii elementów XML – z przestrzeni „gml" i „ng". Rysunek 77 przedstawia trzy sposoby wiązania pojedynczych elementów.

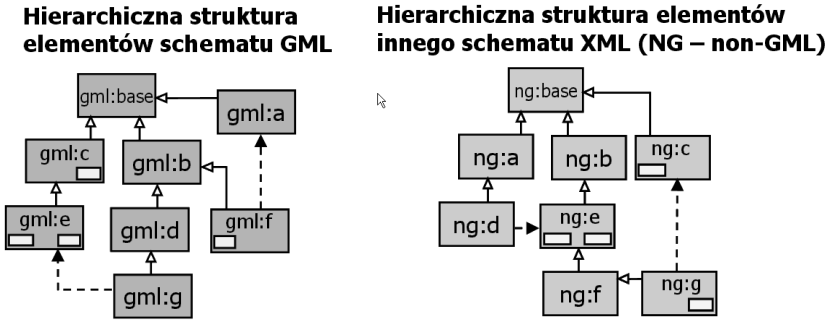
Tabela 3. Lista systemów programowych implementujących GML

Producent oprogramowania:	Nazwy systemów lub programów i typy: (S – serwer, K – klient, S&K – system zawierający i serwer i klienta, E – enkoder)
Implementacje GML 2.0:	
Cogent Logic Corp.	XchainJ (S&K)
CubeWerx Inc.	CubeXPLOR (K), CubeServ Web Map Server (S), CubeServ – Web Feature Server (S), CubeServ – Cascading Web Map Server (S)
Pacific GeoTech Systems Ltd.	shp2gml GeoBroker (S&K)
Implementacje GML 2.1:	
Cadcorp Ltd.	Cadcorp SIS (K)
ESRI	Arc Explorer (K), ArcIMS (S)
Galdos Systems Inc.	Geographic Data Server (GDS) (S), GML Schema Parser (E), FreeStyler (S)
INT Inc.	JCarnacGIS (S&K)
Intergraph Corp.	GeoMedia GML Data Server (K)
Lat-Lon	deegree (S&K)
MapInfo Corp.	MapXtreme Java Ed. (S&K), MapInfo Prof. (E), MapXtend (E), MapMaker J Server (S&K), MapRouting J Server (S&K)
Snowflake Software	GO Loader (K)
Social Change Online Pty Ltd.	MapBroker (K), WebMap Composer (K), WFS-Lite (S)
Implementacje GML 3.0:	
Compusult Ltd.	Web Enterprise Suit (S&K), Map Manager (S&K)
Galdos Systems Inc.	GDS (Geographic Data Server) (S)



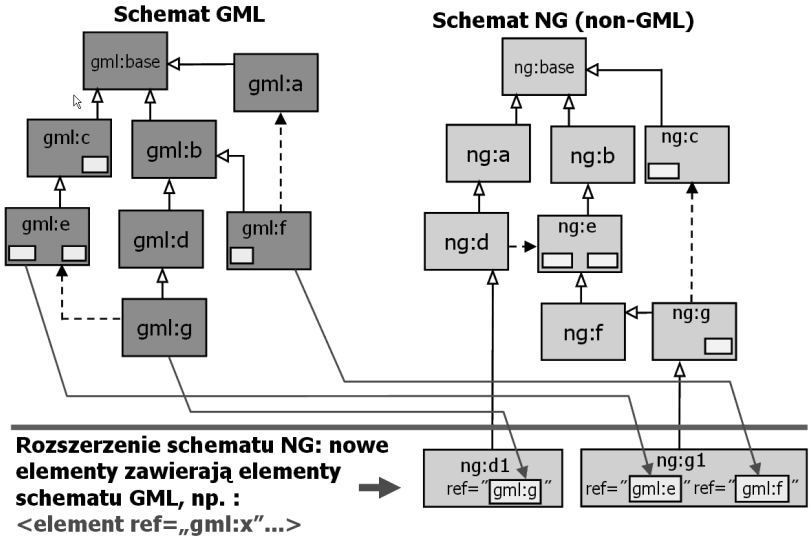
Rys. 77. Łączenie w aplikacjach elementów schematów GML z elementami innych schematów XML.

Dwie oddzielne struktury mogą być użyte do zapisu geoinformacji z określonej dziedziny, lecz brak wzajemnych powiązań tych struktur uniemożliwia wiązanie ze sobą poszczególnych danych. Szczegółowe dane dotyczące jakiejś rzeki nie mogą być związane z wyróżnieniem geoprzestrzennym określającym jej położenie. Rysunek 78 przedstawia takie struktury niepowiązane.



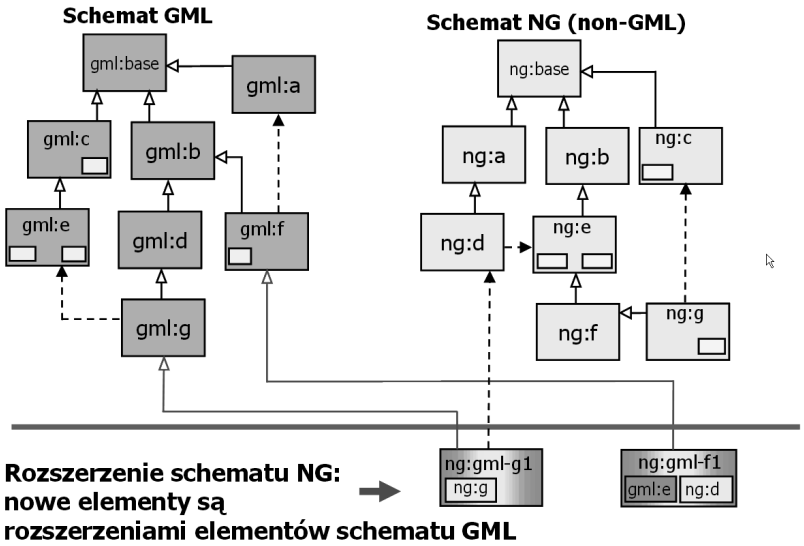
Rys. 78. Dwie oddzielne, niepowiązane struktury w różnych schematach.

Rysunek 79 pokazuje najczęściej stosowane rozwiązanie – nowe elementy schematu (w tym przypadku z przestrzeni tematycznej „ng") zawierają w sobie elementy innego schematu („gml"). Wskazanie, jaki element ma być zawarty, określa argument „ref=".



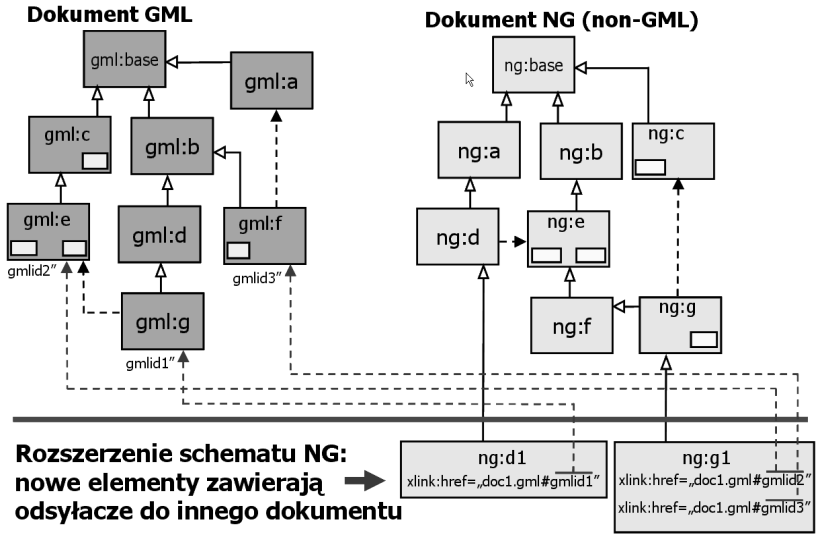
Rys. 79. Łączenie schematów GML z NG przez zawieranie z odwołaniem – „ref=".

Inny sposób łączenia schematów przedstawia rysunek 80. W tym przypadku nowe elementy schematu „ng" są rozszerzeniami elementów innego schematu („gml") i ich związek z pozostałymi elementami schematu „ng" jest realizowany albo przez użycie odsyłacza („xlink:href=") albo przez zawieranie jak w sposobie pierwszym.



Rys. 80. Łączenie schematów GML z NG przez dziedziczenie – „extension".

Trzeci sposób łączenia schematów posługuje się rozszerzeniem języka XML dotyczącym odsyłaczy (xlink) – zagadnienie to jest przedmiotem oddzielnej specyfikacji i daje wiele możliwości odwoływania się do różnych dokumentów rozproszonych w internecie, a także do ich fragmentów lub ściśle określonych miejsc w tych dokumentach. Przykład zastosowania tego rozszerzenia w rozpatrywanym przypadku przedstawia rysunek 81.



Rys. 81. Łączenie schematów GML z NG przez odsyłacze – „xlink:href=".

W wersji 3 języka GML dla realizacji trzeciego sposobu łączenia schematów są zawarte trzy konstrukcje ułatwiające używanie w aplikacjach odsyłaczy przy pomocy „xlink" Konstrukcje te są przedstawione w przykładach 33 do 35.

Przykład 33. [Źródło: specyfikacja GML 3]

```
<attributeGroup name="AssociationAttributeGroup">
      <attributeGroup ref="xlink:simpleLink"/>
      <attribute ref="gml:remoteSchema" use="optional"/>
</attributeGroup>
```

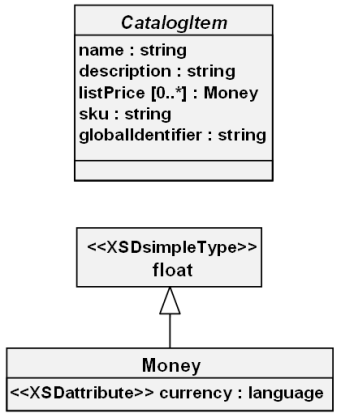
Przykład 34. [Źródło: specyfikacja GML 3]

```
<element name="_reference" type="gml:ReferenceType" abstract="true"/>

<complexType name="ReferenceType">
      <sequence/>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
</complexType>
```

Przykład 35. [Źródło: specyfikacja GML 3]

```
<complexType name="StringOrRefType">
      <simpleContent>
            <extension base="string">
            <attributeGroup ref="gml:AssociationAttributeGroup"/>
      </simpleContent>
</complexType>
```




```
<xs:element name="CatalogItem" type="cml:CatalogItem"/>

<xs:complexType name="CatalogItem">
  <xs:sequence>
    <xs:element name="name" type="xs:string"/>
    <xs:element name="description" type="xs:string"/>
    <xs:element name="listPrice" minOccurs="0">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="cml:Money"
                minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="sku" type="xs:string"/>
    <xs:element name="globalIdentifier" type="xs:string"/>
  </xs:sequence>
</xs:complexType>
```

Rys. 82. Konwersja modeli poprzez „mapowanie" modelu UML do/od XML Schema. [Źródło: (Carslon, 2001)]

5.7.1. Konwersja modeli aplikacyjnych UML do GML 3

Konwersja modeli pojęciowych danych geoprzestrzennych do języka GML jest jednym z istotniejszych i trudniejszych zagadnień technologicznych infrastruktury geoinformacyjnej. Zagadnienie to jest przedstawione w rozdziale 3.6.

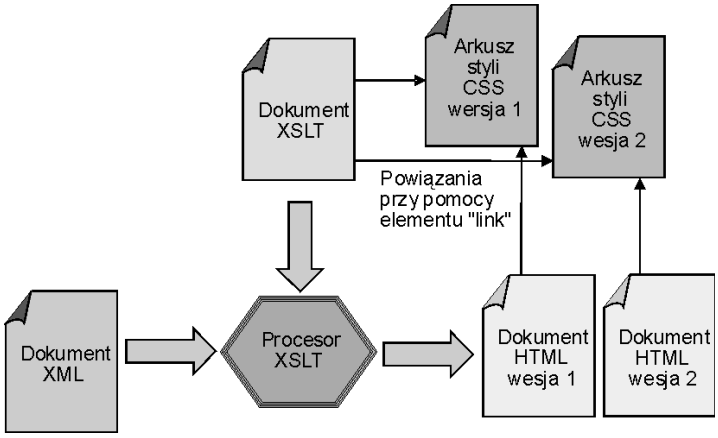
W rozdziałach 3.3.4 i 3.4 przedstawiony jest problem konwersji modeli zapisanych w języku UML do schematów XML za pośrednictwem języka XMI. Rysunek 82 zawiera porównanie diagramu UML i odpowiadającego mu schematu XML. Te ogólne metody mogą być z powodzeniem zastosowane do języka GML jako aplikacji XML.

Jednak postęp prac w tym zakresie jest wyjątkowo duży. Już obecnie istnieje szereg metod dostosowanych specjalnie do zagadnień geoinformacji i języka GML, przykładami mogą być prace prowadzone w amerykańskim NIMA i program SchapeChange opisane w rozdziale 6.4.

5.8. Transformowanie dokumentów GML do innych języków XML

Cenną właściwością języka XML i w konsekwencji jego aplikacji GML jest możliwość przekształcania (transformowania) jego zapisów (dokumentów zapisanych w tym języku). Jednym ze sposobów (opisanym w rozdziale 5.1) jest zastosowanie procesora XSLT (eXtensible Style Language – Transformation). Przy pomocy tego procesora można dokonać konwersji zapisów XML opartych na jednych specyfikacjach do zapisów opartych na innych specyfikacjach – miedzy innymi dla zobrazowania geoinformacji (np. do XHTML lub do SVG). Proces ten ilustruje schemat przedstawiony na rysunku 83. Zastosowanie tej technologii do GML jest obecnie przedmiotem wielu projektów i znalazł między innymi zastosowanie w rozszerzeniach usługi WebMapping opisanej w rozdziale 4.

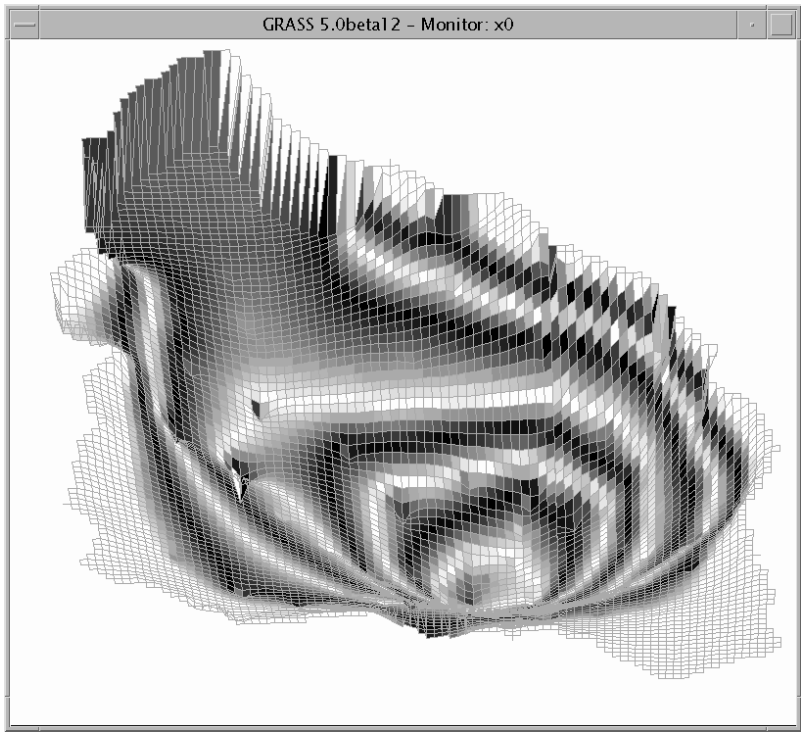
Dokument XML (w tym przypadku GML) może być przy pomocy procesora XSLT przetransformowany na różne dokumenty HTML (lub XML, na przykład SVG) na podstawie różnych arkuszy styli (CSS)



Rys. 83. Generowanie różnych dokumentów XML lub HTML na podstawie jednego dokumentu XML w oparciu o różne arkusze styli.

5.9. Zobrazowanie geoinformacji zapisanej w GML

Przedstawiona powyżej transformacja zapisów w języku GML lub innej aplikacji XML daje możliwość zobrazowania jej w formie wektorowej, na przykład w SVG. Jednak sposób ten jest ograniczony do obrazu dwuwymiarowego (2D). Przykład takiej konwersji zawierają rysunki 41 i 52 w rozdziale 4. W wielu przypadkach takie zobrazowanie nie jest wystarczające i poszukuje się sposobów przestrzennego przedstawienia geoinformacji zapisanej z uwzględnieniem wszystkich wymiarów – na co pozwala 3. wersja GML. Przykład przestrzennego zobrazowania informacji geologicznej przedstawia rysunek 84.



Rys. 84. Przestrzenne zobrazowanie pokrycia rastrowego (macierzowego) z uwzględnieniem trzeciego wymiaru w systemie GRASS.