POLSKIE
TOWARZYSTWO
INFORMACJI
PRZESTRZENNEJ

ROCZNIKI GEOMATYKI 2012

Modele danych przestrzennych w UML i ich transformacja do schematów GML i struktur baz danych

Tom X
Zeszyt 1(51)
Warszawa

POLSKIE TOWARZYSTWO INFORMACJI PRZESTRZENNEJ
ROCZNIKI GEOMATYKI 2012 ❍ TOM X ❍ ZESZYT 1(51)

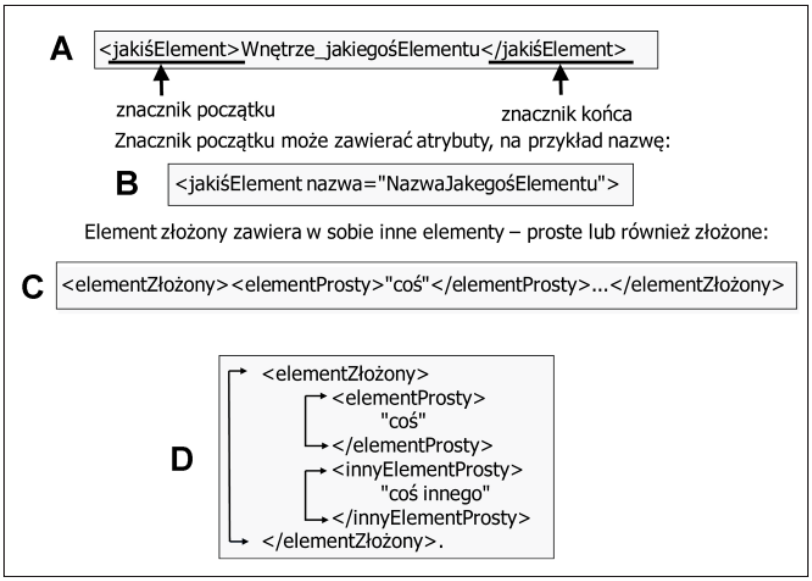
Janusz Michalak

2. Różnice pomiędzy językiem zapisu danych i jego dziedzinową aplikacją

Bardzo często, bez głębszego zastanowienia się, język GML i inne języki z nim powiązane nazywa się formatem zapisu danych przestrzennych. Jednak głębsza analiza tego zagadnienia wykazuje, że jest olbrzymia różnica pomiędzy taką formą zapisu a sztywnymi formatami stosowanymi dawniej i jeszcze obecnie. Z tego względu w rozdziale tym opisane są podstawy zapisu znacznikowego w oparciu o język XML, który jest fundamentem dla języka GML. Rozdział zawiera także krótką historię zapisu geoinformacji, relacje pomiędzy GML a innymi językami od niego pochodnymi lub z nim powiązanymi, a także informacje na temat dalszych prac nad rozwojem GML.

2.1. Podstawy zapisu znacznikowego na bazie języka XML

Zarówno zapisy w GML, jak i w innych językach znacznikowych bazujących na XML, są zwykłym prostym tekstem, często niesłusznie nazywanym formatem ASCII. ASCII to sposób kodowania znaków, który ze względu na jego ograniczenia jest tu zastąpiony przez standard Unicode w wersji UTF-8. Tekst ten jest podzielony na fragmenty (nazywane elementami) przy pomocy znacznika początku elementu w postaci `<nazwaTypuElementu>` i znacznika końca różniącego się znakiem ukośnika przed nazwą (rys. 2.1A).



Rys. 2.1. ABC języka XML. Jest to zwykły tekst podzielony na fragmenty za pomocą znaczników (Michalak, 2003b). Objaśnienia w tekście.

Element może być prosty i taki nie zawiera wewnątrz innych elementów a jedynie tekst lub może być złożony – może zawierać w sobie wiele innych elementów (rys. 2.1C). Jeżeli one są również złożone mamy do czynienia z wielokrotnym zagnieżdżeniem, które w szczególnych przypadkach może być nieskończone (rys. 2.16 w rozdziale 2.5). Element może posiadać atrybuty, które odnoszą się do wnętrza elementu – do jego zawartości. Atrybuty są umieszczane w znaczniku początku w formie `nazwaAtrybutu=„wartośćTegoAtrybutu”` (rys. 2.1B). Poniżej przedstawiony jest przykład (2.1) zastosowania atrybutu (tekst pogrubiony) do określenia języka w jakim jest napisany tekst wewnątrz elementu:

Przykład 2.1. Elementy tekstowe z podaniem języka w atrybucie elementy.

```
<gmd:PT_FreeText>
  <gmd:textGroup>
    <gmd:LocalisedCharacterString locale="#locale-eng">Warsaw</gmd:LocalisedCharacte
    <gmd:LocalisedCharacterString locale="#locale-pol">Warszawa</gmd:LocalisedCharac
  </gmd:textGroup>
</gmd:PT_FreeText>
```

Dla lepszej czytelności poszczególne składniki zapisu dzieli się na linie tekstu ze stopniowo powiększającym się wcięciem dla pokazania hierarchicznego zagnieżdżania się elementów (rys. 2.1D i 2.2). Jest to jednak potrzebne jedynie człowiekowi, ponieważ system programowy, który generuje takie zapisy, czyta je lub przetwarza nie potrzebuje takiego układ tekstu dla poprawnej analizy zapisu.

```
<?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>
```

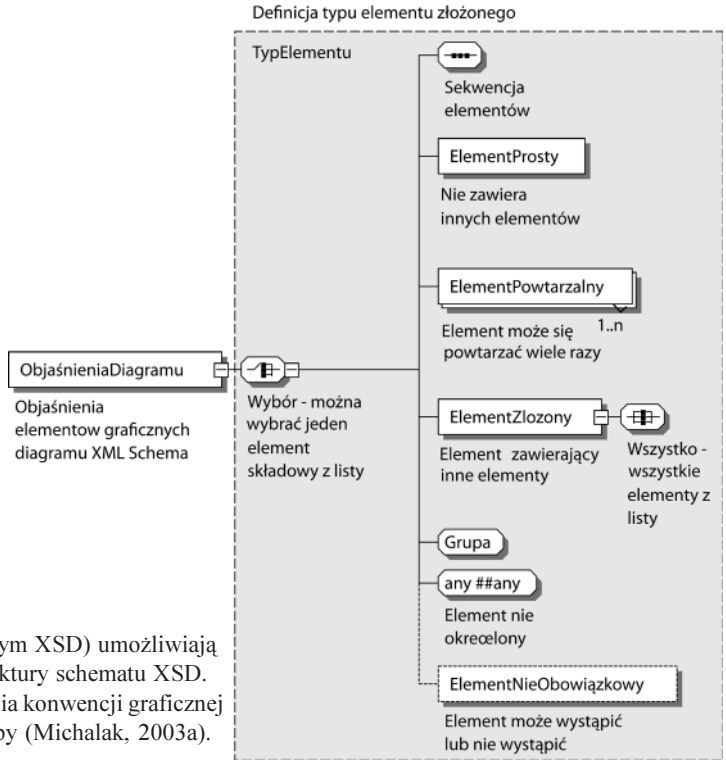
Rys. 2.2. Reguły zapisu dokumentu XML są określone w schemacie XSD, który również jest zapisany w ten sam sposób (Michalak, 2003b). Przykład uproszczony.

W pliku tekstowym (dokumencie) można zapisywać dane z określonej dziedziny tylko przy pomocy zdefiniowanej listy typów elementów (zdefiniowanego słownika). Zbiór tych definicji jest zapisany w tak zwanym schemacie XSD *(XML Schema Definition)*. Bardzo prosty i niepełny przykład schematu XSD dla zapisu okresu czasu jest przedstawiony na rysunku 2.2, przykład takiego zapisu zgodnego z tym schematem przedstawia rysunek 2.3.

Rys. 2.3. Przykład zapisu XML zgodny ze schematem XSD przedstawionym na rys. 2.2 (Michalak, 2003b). Przykład uproszczony.

```
<?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>
<!– ... –>
```

Ponieważ zapisy te są zwykłym tekstem, można je pisać przy pomocy najprostszego edytora teksu, na przykład przy pomocy notatnika *(notepad)*. Jest to jednak trudne i stwarza możliwość popełniania wielu błędów. Z tego względu wskazane jest posługiwanie się przeznaczonymi do tego specjalistycznymi edytorami. Przykłady takich edytorów lub fragmentów ich okien przedstawione są na rysunkach 2.9, 2.11, 2.15, 2.16, 9.3, 11.9 i 11.10. Wiele z nich pozwala na graficzną analizę i edycję dokumentów XML. Rysunek 2.4 objaśnia składniki diagramu stosowanego w edytorze XML Spy firmy Altova. Przykład innego diagramu elementów stosowanego w edytorze Oxygen (o nazwie w formie znacznika `<oXygen/>`) firmy Syncro Soft jest przedstawiony na rysunku 2.10.



Rys. 2.4. Edytory XML (i w tym XSD) umożliwiają graficzne przedstawienie struktury schematu XSD. Rysunek przedstawia objaśnienia konwencji graficznej przyjętej w edytorze XML Spy (Michalak, 2003a).

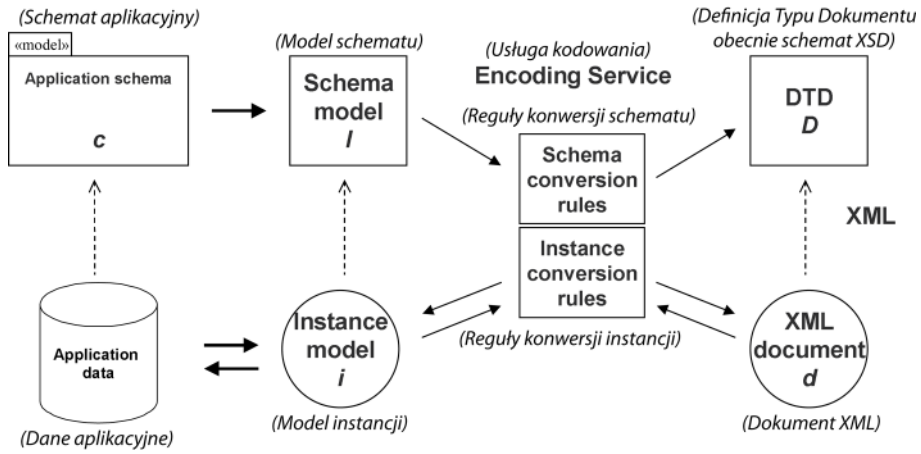
2.2. Wprowadzenie do języka GML

U podstaw koncepcji i aktualnie prowadzonych prac nad GML leży szereg wstępnych założeń:

- język powinien dać możliwość efektywnego zapisu w celu przesyłania geoinformacji w heterogenicznym rozproszonym środowisku systemowym;
- język powinien umożliwić przesyłanie tej informacji niezależnie od stopnia agregacji – zarówno pojedyncze wyróżnienia jak i duże kolekcje danych, na przykład mapy numeryczne;
- informacja zapisywana w GML powinna dotyczyć całego zakresu danych – metadanych, opisu elementów geometrycznych wraz ze współrzędnymi określającymi te elementy, rodzaje zastosowanych układów odniesienia i odwzorowania wraz ze wszystkimi parametrami tych układów, podstawowe atrybuty wyróżnień, a także informację o sposobie łączenia wyróżnień w zbiory wyróżnień i o powiązaniach z innymi rodzajami informacji;
- poszczególne wyróżnienia (nawet w obrębie jednego zbioru wyróżnień) mogą być odniesione geoprzestrzennie w różnych układach;
- geoinformacja powinna być zapisana niezależnie od skali ewentualnej późniejszej wizualizacji (na ekranie lub na papierze) – jednak może być powiązana z innymi dokumentami (zapisami tekstowymi) określającymi graficzne formy przeznaczone do jej wizualizacji, odpowiednimi do jej treści, a także skali.

W trakcie rozwoju języka GML (rys. 2.12) okazało się, że zakres jego zastosowań może być znacznie szerszy niż początkowo zakładano. Język ten może być z równym powodzeniem stosowany także do przechowywania danych w bazach i jako rodzimy (wewnętrzny) format w systemach GIS do przetwarzania, analizy i filtrowania geoinformacji. Łatwość modyfikowania go i łączenia z innymi językami wyprowadzonymi z XML daje duże możliwości opracowywania wyspecjalizowanych odmian tego języka dla specyficznych zastosowań w różnych dziedzinach posługujących się geoinformacją. Przykłady takich wyspecjalizowanych rozszerzeń GML są przedstawione w rozdziale 2.4.

Dlaczego język, a nie format? Istnieje przecież wiele formatów do zapisu geoinformacji w przeróżnych postaciach, czy rzeczywiście potrzebne jest jeszcze coś nowego? Doświadczenia wynikające z zastosowań zaawansowanych złożonych systemów geoinformacyjnych i nowych technologii przesyłania danych wykazują jednak, że dotychczasowe rozwiązania oparte na sztywnych formatach bazujących na zamkniętych prostych modelach danych geoprzestrzennych nie są wystarczające. Podobne zjawiska obserwuje się także w innych dziedzinach w odniesieniu do innych rodzajów informacji. W przeciwieństwie do formatu, język – a w tym przypadku GML – pozwala na dynamiczne zapisywanie danych w różnej formie zależnej od zawartości tych danych i od aktualnej potrzeby. W rezultacie przy pomocy języka można tą samą geoinformacją zapisać różnie i nie jest to wada takiego podejścia, lecz zaleta. Tą zaletę ilustruje przykład schematu dynamicznego zapisu danych geoprzestrzennych opartego na zapisie modelu danych w języku UML, konwersji tego modelu do zapisu w języku XMI i wygenerowania na tej podstawie schematu XSD dla modyfikacji zapisu danych w języku GML. Schemat ten jest opisany w pracy D. Skogana (1999). Rysunek 2.5 przedstawia taki schemat dynamicznego użycia tych języków do przekazu geoinformacji.

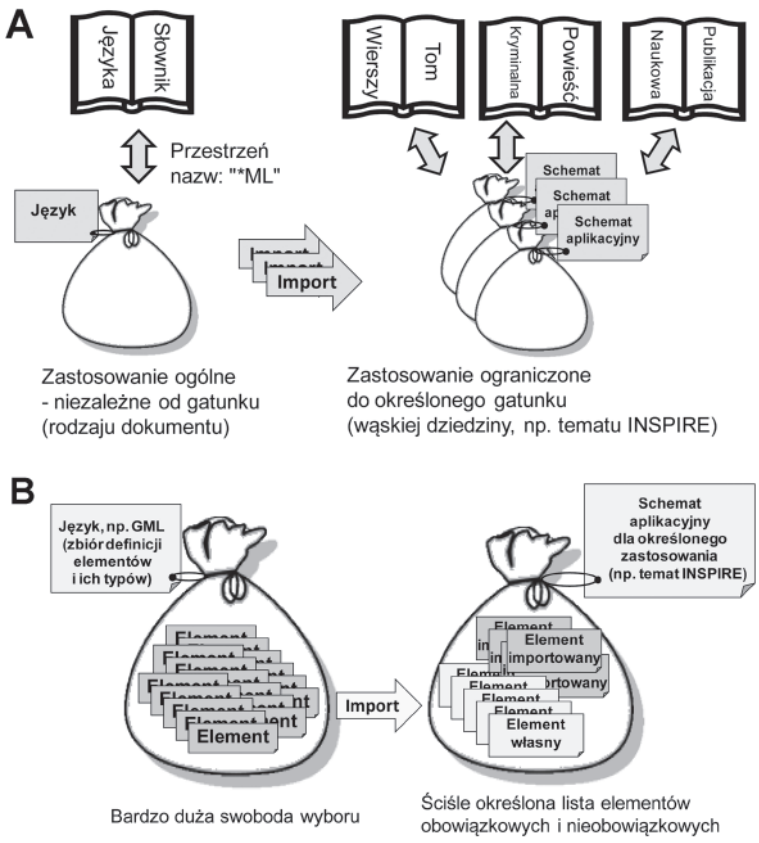


Rys. 2.5. Ogólny schemat procesu transformacji danych do XML z zastosowaniem modeli UML (Skogan, 1999).

Słowo „język" kojarzy się nam najczęściej z językiem naturalnym. Jednak we współczesnej informatyce stosowana jest olbrzymia liczba przeróżnych wyspecjalizowanych notacji nazywanych również językami. Najogólniej języki te można podzielić na cztery grupy:

1. Języki programowania – język programowania składa się z dwóch części, pierwsza to reguły syntaktyczne i druga to semantyka. Obie te części określają, jak należy pisać poprawne wyrażenia i jak mają być interpretowane przez kompilatory tych języków.
2. Języki komunikatów interfejsowych – do nich należą języki zapytań (na przykład SQL i OQL), a także języki poleceń, na przykład w protokóle HTTP składnia poleceń Get i Post jest używana w standardowych usługach sieciowych dotyczących danych geoprzestrzennych (CSW, WMS, WFS i innych).
3. Języki specyfikacyjne – w przypadku danych są to języki określające struktury tych danych i należą do nich DDL, ODL, XSD i także w pewnym stopniu UML, OIL *(Ontology Inference Layer* lub *Ontology Interchange Language)* i inne z nim powiązane.
4. Języki znacznikowe – dzielą się na trzy kategorie: prezentacyjne, proceduralne i opisowe. Zapis w znakowaniu opisowym nazywany także semantycznym i dzieli tekst na fragmenty znaczeniowe. Obecnie olbrzymia liczna różnych dziedzinowych języków opisowych stosujących znakowanie ogólne *(generic markup)* oparta jest na standardzie języka (lub raczej metajęzyka) XML.

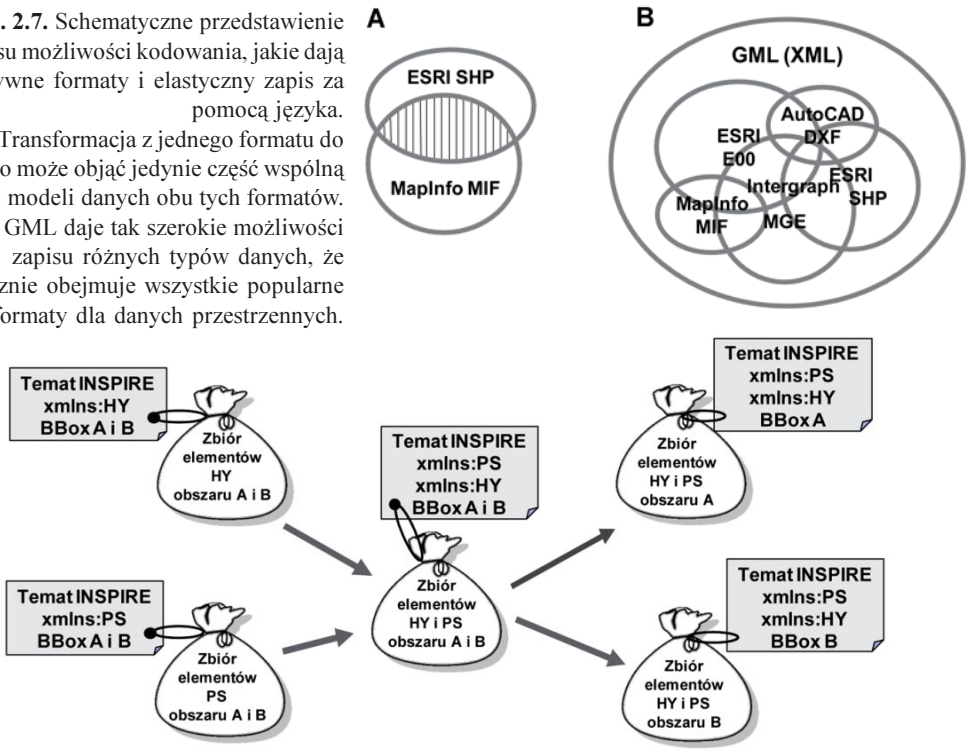
Pomiędzy językami naturalnymi i językami stosowanymi w informatyce występuje wiele podobieństw. Można tu użyć metafory ontologicznej porównując między sobą różne komponenty występujące w obu kategoriach. Syntaktyka w językach informatycznych odpowiada w przybliżeniu regułom gramatycznym języka naturalnego, a semantyka języka informatycznego określa znaczenie poszczególnych elementów, co w języku naturalnym odpowiadałoby roli słownika (rys. 2.6).



Rys. 2.6.
A – Metaforyczne porównanie języka XML i jego dziedzinowych aplikacji do języka naturalnego – objaśnienia w tekście. B – Język (zarówno naturalny jak i XML) jest miedzy innymi „słownikiem" (magazynem elementów). Z tego magazynu wybiera się do schematu aplikacyjnego te elementy, które są potrzebne w określonej dziedzinie zastosowań.

Dokument XML (w tym także GML) zawierający dane można porównać z książką, a język aplikacyjny zdefiniowany w schemacie XSD, w jakim ten dokument jest zapisany, można porównać ze słownikiem języka naturalnego. Jednak jest tu istotna różnica – powiązanie dokumentu (książki) z różnymi schematami (słownikami) wymaga precyzyjnego określenia za pomocą elementu `import`, a każdy „zaczerpnięty" element (słowo) jest poprzedzone przedrostkiem określającym przestrzeń nazw (zakres słownika) i wskazującym z jakiego schematu (słownika) pochodzi definicja tego elementu (słowa). Element (instrukcja) `import` powoduje przyłączenie „słownika" do schematu aplikacyjnego, co daje możliwość wykorzystywania w schemacie elementów zdefiniowanych w tym słowniku. W takim przypadku elementy importowane mają inną przestrzeń nazw niż przestrzeń aplikacji. Element (instrukcja) `include` jest używana do łączenia poszczególnych dokumentów XSD w ramach jednego schematu aplikacyjnego. Ma to zastosowanie, gdy elementy tego schematu zdefiniowane w jednym dokumencie są używane w innych. W takim przypadku najczęściej wszystkie te elementy mają jedną wspólną przestrzeń nazw.

Zastosowanie języka GML do zapisu danych geoprzestrzennych daje dużą przewagę nad zapisami przy użyciu tradycyjnych formatów. Zakres zastosowań tego języka jest znacznie szerszy (rys. 2.7). Format tekstowy daje dużą swobodę w manipulowaniu zapisem. Pliki z zapisami danych w GML, gdy przestrzegane są reguły dotyczące schematów aplikacyjnych,



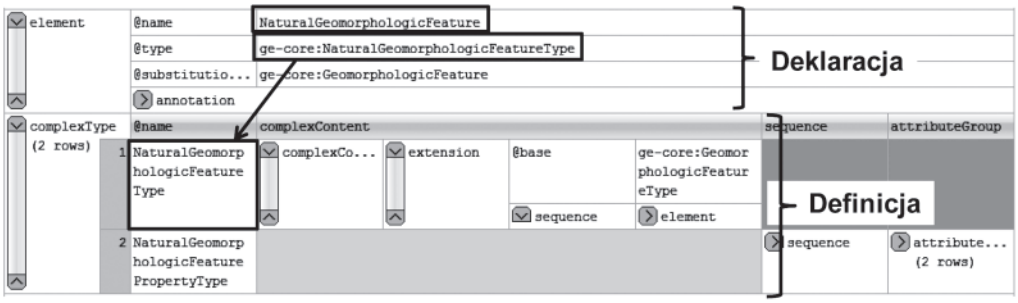
Rys. 2.7. Schematyczne przedstawienie zakresu możliwości kodowania, jakie dają sztywne formaty i elastyczny zapis za pomocą języka.
A – Transformacja z jednego formatu do innego może objąć jedynie część wspólną modeli danych obu tych formatów.
B – Język GML daje tak szerokie możliwości zapisu różnych typów danych, że praktycznie obejmuje wszystkie popularne formaty dla danych przestrzennych.

Rys. 2.8. Przykłady łączenia i rozdzielania zbiorów danych zapisanych za pomocą języka GML.

można dowolnie w prosty sposób (np. edytorem tekstu) łączyć lub dzielić, jednak z zachowaniem kilku warunków, w tym deklaracji przestrzeni nazw – `xmlns` i prostokąta ograniczającego – `BBox` (rys. 2.8).

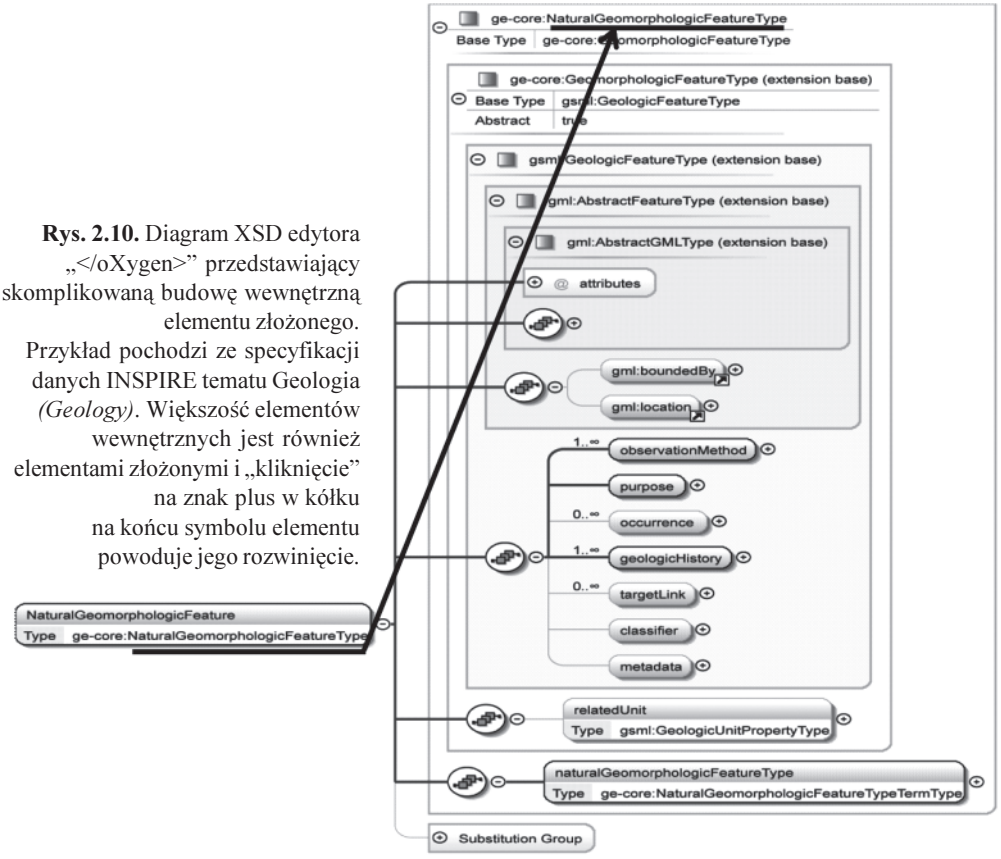
W jednym zbiorze danych zapisanym zgodnie z regułami XML można umieszczać elementy z różnych języków dziedzinowych, zarówno dla danych przestrzennych (w GML i językach od niego pochodnych), jak i danych nieprzestrzennych zawartych w elementach schematów niezależnych od GML. Można również dowolne elementy zdefiniowane w schematach GML używać w innych językach. Jednak takie postępowanie wymaga dużego doświadczenia i zakłada z góry, że typowe oprogramowanie dedykowane językowi GML nie będzie mogło poprawnie interpretować takiego zapisu.

Rysunek 2.9 przedstawia specyfikację elementu ze schematu aplikacyjnego dla tematu Geologia *(Geology)* dyrektywy INSPIRE. Specyfikacje takie składają się z dwóch części – deklaracji, w której określa sie nazwę i wskazanie na typ określony w drugiej części nazywanej definicją typu. Definicja typu szczegółowo określa zawartość elementu. W przypadku elementów złożonych zawartość ta może składać się z wielu elementów, które mogą być hierarchiczne zagnieżdżone. Typy elementów składowych mogą być zdefiniowane w tym samym schemacie (w tym samym pliku lub pliku przyłączonym instrukcją `include`). Syntaktyka języka XML pozwala na użycie więcej niż jednej deklaracji odwołującej się do jednej definicji. W rezultacie może być kilka elementów o różnych nazwach tego samego typu – o takiej samej budowie wewnętrznej określonej w definicji typu.



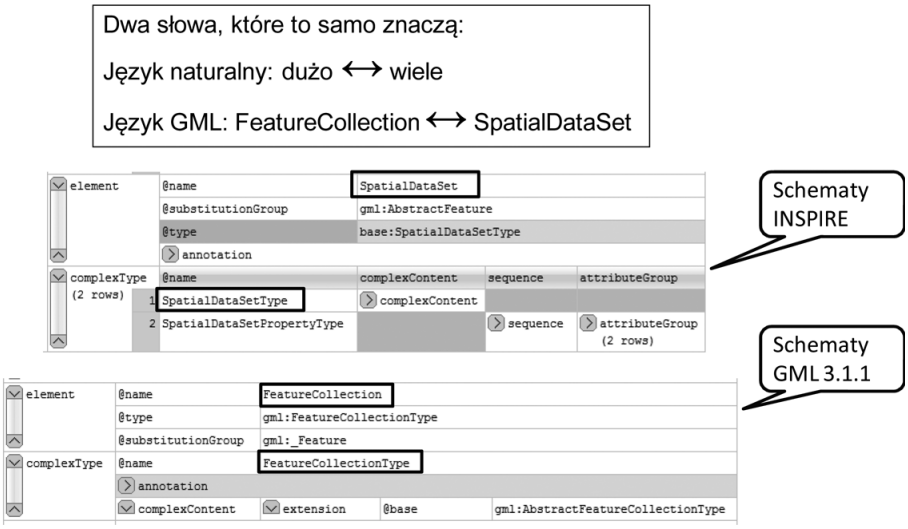
Rys. 2.9. Każdy element w schemacie aplikacyjnym opartym na GML jest specyfikowany dwukrotnie: jako jego deklaracja i jako definicja jego typu. Tabelaryczne przedstawienie elementów schematu w oknie edytora XML o nazwie </oXygen>.

Rysunek 2.10.przedstawia graficzny schemat XSD (w konwencji zobrazowania przyjętej w edytorze XML „</oXygen>") złożonego elementy o skomplikowanej budowie Na uwagę zasługuje fakt, że diagramy graficzne tego edytora są również zapisane wektorowo przy pomocy języka SVG *(Scalable Vector Graphics)* przeznaczonego do zapisu takiej grafiki. Dzięki takiemu rozwiązaniu zobrazowanie informacji XML jest realizowane też za pomocą tego języka, a ściślej – innej jego aplikacji.



Rys. 2.10. Diagram XSD edytora „</oXygen>" przedstawiający skomplikowaną budowę wewnętrzną elementu złożonego. Przykład pochodzi ze specyfikacji danych INSPIRE tematu Geologia *(Geology)*. Większość elementów wewnętrznych jest również elementami złożonymi i „kliknięcie" na znak plus w kółku na końcu symbolu elementu powoduje jego rozwinięcie.

Kolejnym przykładem analogii pomiędzy językami znacznikowymi a językiem naturalnym są synonimy – różne określenia mają to samo znaczenie. W języku GML zapis danych w jednym pliku składa się ze zbioru elementów typu `FeatureMember`. Cały ten zbiór jest umieszczony w jednym elemencie pojemniku (stosowane jest także określenie korzeń – *root*) i w różnych aplikacjach nazwa tego pojemnika może być różna, na przykład specyfikacja usługi WFS *(Web Feature Service)* wymaga, aby nazywał się on `FeatureCollection`, a dla danych INSPIRE jest wymagana nazwa `SpatialDataSet` (rys. 2.11).



Rys. 2.11. Synonimy językowe dotyczące nazw głównych elementów (pojemników – określanych także jako *root*). W górnym przykładzie dotyczy to danych INSPIRE, a w dolnym specyfikacji WFS i wersji 3.1.1 języka GML.

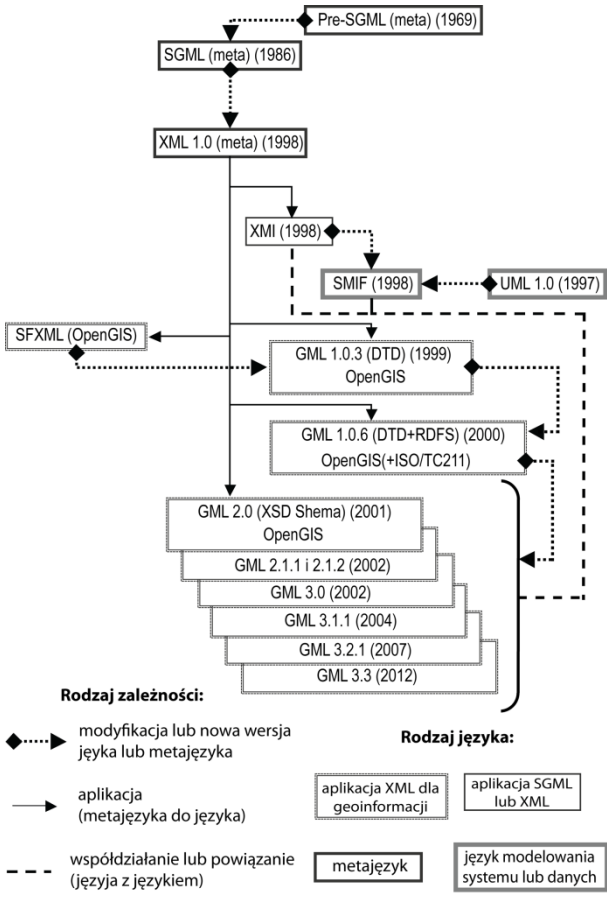
Przedstawione w tym rozdziale krótkie wprowadzenie do języka GML porusza tylko kilka wybranych najważniejszych ogólnych aspektów tego zagadnienia. Nie wystarczy to do pełnego zrozumienia zasad opracowania schematów aplikacyjnych i praktycznego ich wykorzystania. Z tego względu warto na zakończenie wskazać bardziej obszerne źródła informacji z tego zakresu. Obok książki poświęconej temu językowi (Lake i inni, 2004) wyróżnić należy:

1) łatwo dostępne specyfikacje OGC *(Open Geospatial Consortium),* w tym między innymi:
   - GML Extended schemas and encoding rules, v. 3.3.0 (Portele, 2012),
   - OpenGIS Geography Markup Language (GML) Encoding Standard, v. 3.2.1 (Portele, 2007), znany także jako norma ISO 19136:2007 (ISO/TC 211, 2007a),
   - Geography Markup Language (GML) simple features profile (with Corrigendum), v. 2.0 (Brink, Portele, Vretanos, 2011);
2) publikacje naukowe i popularnonaukowe dostępne w Internecie, na przykład:
   - The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS) (Peng, Zhang, 2004),
   - GML-Based Interoperable Geographical Databases (Zhang i inni, 2003),
   - Building GML-native web-based geographic information systems (Huang i inni, 2009),
   - Visualization of GML data using XSLT (Tennakoon, 2003).

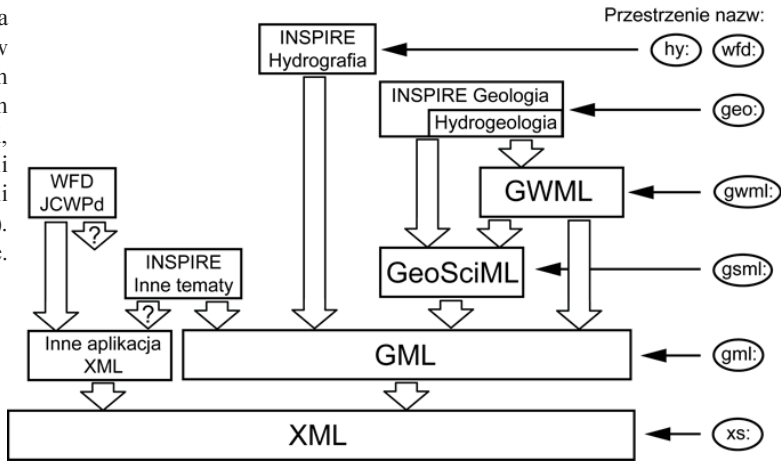
Przedstawione powyżej pozycje to tylko przykłady wybrane z wielkiej liczby publikacji dostępnych w formie elektronicznej.

2.3. Krótka historia zapisu geoinformacji

Problem zapisu danych przestrzennych powstał równocześnie z pierwszymi systemami typu GIS w połowie lat sześćdziesiątych XX w. W pierwszym okresie formy zapisu tych danych były ściśle związane z konkretnymi systemami programowania i najczęściej były to zapisy binarne. Jednak potrzeba wymiany danych pomiędzy różnymi systemami sprawiła, że zaczęto opracowywać formaty niezależne od systemów, na przykład: POLYVRT (1974), ODYSSEY (1976). Następny krok w tym zakresie to standardowe formaty w Stanach Zjednoczonych dla celów statystycznych (DIME i TIGER). Kolejne lata to dominacja firm Intergraph i ESRI w zakresie zapisu danych, co doprowadziło do powstania dojrzałych rozwiązań, ale ograniczonych tylko do ich własnych koncepcji technologicznych. Lata dziewięćdziesiąte to z kolei burzliwy rozwój formatów narodowych i dalsze pogłębianie sie problemów z wymianą danych pomiędzy coraz większą liczbą różnorodnych systemów. Tendencja do nieujawniania formatów wewnętrznych przez czołowe firmy z tego zakresu doprowadziła w połowie lat dziewięćdziesiątych do impasu w pracach OGC nad interoperacyjnością. Jednak kryzys ten został przełamany i w roku 1998 Ron Lake zainicjował prace w OGC nad językiem GML. Był to radykalny zwrot w zakresie zapisu danych geoprzestrzennych, ponieważ zastosowanie zapisu znacznikowego XML pozwoliło na wprowadzenie niezbędnej elastyczności i rozszerzalności sposobu zapisu różnorodnych rodzaj danych stosowanych w różnych dziedzinach. Prekursorem całej rodziny języków GML był język SFXML *(Simple Features XML)* opracowany i opublikowany w roku 1999. Od tej pory znacznikowy zapis danych geoprzestrzennych (2.12) jest dominujący i kolejne lata przynoszą nowe bardziej dojrzałe i bardziej rozbudowane jego wersje. Na bazie tego języka powstało od tego czasu wiele aplikacji dziedzinowych, a także dziedzinowych języków pochodnych, które również służą jako baza dla dalszych aplikacji (rys. 2.13).



Rys. 2.12. Drzewo genealogiczne języka GML



Rys. 2.13. Hierarchiczna struktura języków i nierozszeralnych schematów aplikacyjnych dla nauk o Ziemi, w tym geologii i hydrogeologii (Michalak i in., 2011). Objaśnienia w tekście.

Jednak w tym szybkim i burzliwym rozwoju języka GML można zaobserwować niepokojące zjawiska. Ponownie pojawił się problem z rozwiązaniami narodowymi, szczególnie w zakresie danych topograficznych. Z tego powodu interoperacyjne łączenie danych topograficznych z sąsiednich krajów jest bardzo trudne, a czasem niemożliwe. Można z przekąsem powiedzieć, że dzięki ujednoliconej symbolice kartograficznej „interoperacyjność" tradycyjnych map papierowych była większa, ponieważ mapy z krajów sąsiadujących można było z powodzeniem sklejać. Inny problem związany z aplikacjami dziedzinowymi GML to opracowywanie w różnych ośrodkach schematów aplikacyjnych dla tych samych zastosowań. To również prowadzi do problemów z zakresu interoperacyjności oraz wykazuje niedojrzałość w podejmowaniu takich przedsięwzięć.

2.4. Języki oparte na GML i z nim powiązane

Cechą specyficzną dla aplikacji XML, a w tym przypadku bazujących także na języku GML, jest możliwość tworzenia hierarchicznych struktur aplikacyjnych. Rysunek 2.13 przedstawia przykład takiej struktury z zakresu danych w obszarze nauk o Ziemi. Kolejne schematy aplikacyjne są coraz bardziej wyspecjalizowane i dedykowane coraz węższemu zakresowi tematycznemu. Bazą są w takich przypadkach języki (na przykład XML i GML), a na samej górze występują ściśle ograniczone i jednoznacznie wyspecyfikowane schematy aplikacyjne do dokładnie określonych zastosowań, najczęściej już dalej nierozszerzalne. W takiej strukturze hierarchicznej często jest trudno dokładnie określić, co jest jeszcze językiem, a co już nierozszerzalnym schematem aplikacyjnym.

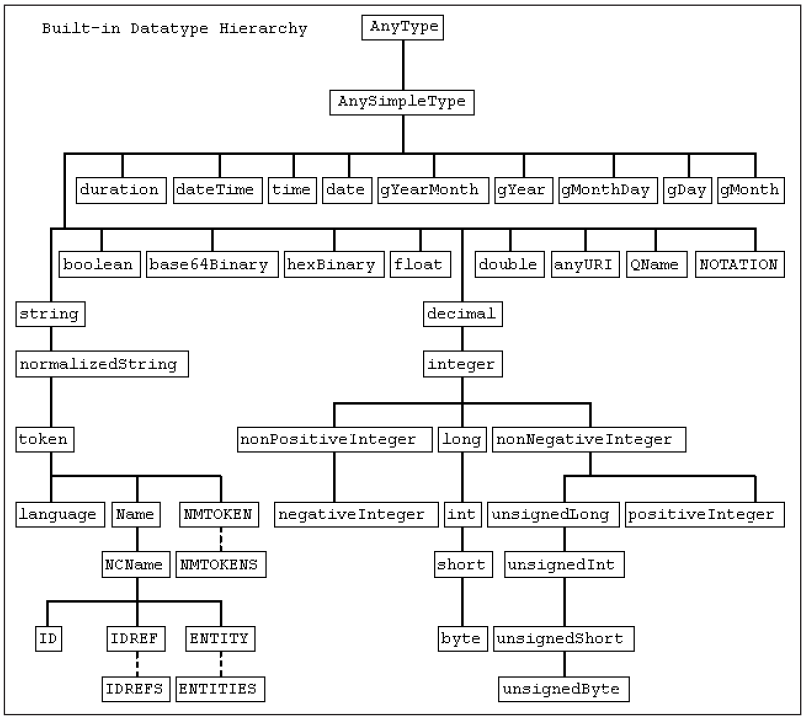
Na rysunku 2.13 strzałki pionowe określają zależności pomiędzy poszczególnymi aplikacjami, na przykład język GWML (Boisvert, Brodaric, 2008) wykorzystuje elementy zdefiniowane w języku GeoSciML (IGW-CGI-IUGS, 2008) i w języku GML (Michalak, 2005b). Strzałki poziome przypisują poszczególnym aplikacjom przestrzenie nazw elementów, na przykład schematy specyfikacji danych tematu Geologia *(Geology)* dyrektywy INSPIRE definiują elementy z przestrzeni „`geo:`".

Jedną z generalnych zasad dotyczących języków jest pozostawianie dużej swobody w określaniu typów elementów, dla których są zdefiniowane hierarchiczne struktury dziedziczenia. Może to na przykład dotyczyć wektorowych atrybutów geometrycznych dla wyróż-

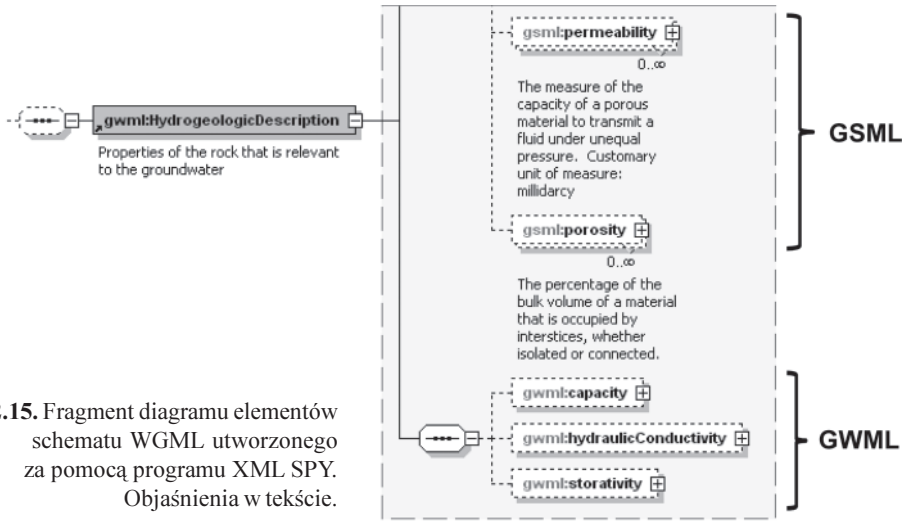
nień *(feature)* i przykład takiej struktury przedstawiony jest w rozdziale 9.1 (rys. 9.2) lub atrybutów geometrycznych dla pokryć *(coverage)*, co przedstawia rysunek 9.5 w rozdziale 9.2. W takich przypadkach w języku powinno się specyfikować typ położony możliwie najwyżej w drzewie hierarchicznym, aby w schemacie aplikacyjnym można było wybrać stosowny w danym przypadku i najczęściej jedyny typ leżący poniżej. Inny przykład pozostawienia dużej swobody to proste typy danych określane dla elementów prostych. Ponieważ język jest w pewnym sensie szablonem dla opracowywania ścisłych aplikacji, zadeklarowanie typu prostego jako `Any` (dowolny) lub `AnyType` daje dużą swobodę w dostosowaniu tego elementu do określonych potrzeb aplikacyjnych przez podstawienie dowolnego typu prostego położonego niżej w hierarchii, na przykład:

- `characterString` – dla dowolnego ciągu znaków, najczęściej kodu lub tekstu bez określenia języka,
- `localizedCharacterString` – dla tekstu z określeniem języka w atrybucie,
- `PT_FreeText` – dla sekwencji elementów typu `localizedCharacterString`, gdy jest to w kilku językach (przykład 2.1 w rozdziale 2.1),
- mogą to także być dowolne typy liczbowe: `integer, unsignedInteger, float` lub `double`, a także bardziej ogólny `decimal`,
- również inne proste typy, jak `date, dateTime` lub `boolean`.

Przykład hierarchii prostych typów zdefiniowanych *(built-in)* w języku XML przedstawia rysunek 2.14.



Rys. 2.14. Hierarchia prostych typów wbudowanych *(built-in)* w języku XML (Biron, Permanente, Malhotra, 2004). Objaśnienia w tekście.



Rys. 2.15. Fragment diagramu elementów schematu WGML utworzonego za pomocą programu XML SPY. Objaśnienia w tekście.

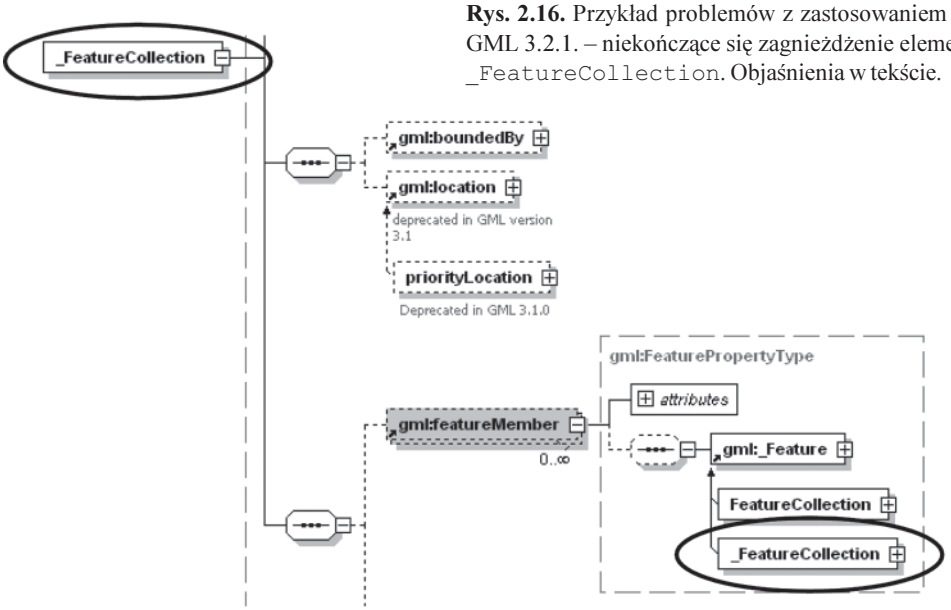
W konkretnej specjalizowanej do określonych potrzeb aplikacji stosowanie typów ogólnych jest niepoprawne, ponieważ powoduje to niejasność i brak precyzji, co z kolei prowadzi do dowolności wyboru typu w zapisach danych określonych schematem aplikacyjnym i jest przyczyną błędów. Znacznie utrudnia to lub uniemożliwia interpretację takich zapisów i stwarza poważne problemy z zakresu interoperacyjności.

Jest także szereg innych przypadków, w których obowiązują inne zasady dla języka i inne dla specjalizowanej aplikacji. Między innymi dotyczy to stosowania w językach szablonów *(templete)*, na przykład w języku GeoSciML do takich szablonów należą: `ScopedName`,`ControlledConcept`, `LocalizableGenericName` i `CGI_Term` (rys. 11.8 w rozdziale 11).

Możliwość korzystania z elementów zdefiniowanych w różnych językach w specjalizowanej dla danych zastosowań aplikacji jest bardzo cenną zaletą zapisu znacznikowego. Pozwala to na łączenie typów elementów (pozycji słownikowych) z różnych dziedzin w jednej aplikacji, która przez to może mieć charakter interdyscyplinarny. Przykład takiego łączenia przedstawia rysunek 2.15. Element złożony `HydrogeologicalDescription` zawiera elementy z dwóch języków: `gsml:permeability` i `gsml:porosity` z języka GeoSciML oraz również `gwml:capacity`, `gwml:hydraulicConductivity` i `gwml:storativity` z języka GroundWaterML.

2.5. Przyszłość języka GML

W rozdziale 2.3 przedstawiono historię języka GML. Kolejne wersje przedstawione na rysunku 2.12. były coraz bardziej dojrzałe, rozbudowane i uniwersalne. Pozwalały na zapis prawie wszystkich typów danych określonych w normach grupy ISO 19100 z uwzględnieniem specyficznych wymogów różnorodnych dziedzin zastosowań. W pracach nad rozwojem tego języka przykładano szczególną wagę do tego, aby był on rozszerzalny i elastyczny. Jednak, aby mógł być skutecznie implementowany na różnych platformach konieczne było określenie wielu ograniczeń przedstawionych w aneksie Do normy ISO 19136, specyfikującym reguły transformacji modelu UML do schematów XSD.



Rys. 2.16. Przykład problemów z zastosowaniem języka GML 3.2.1. – niekończące się zagnieżdżenie elementów `_FeatureCollection`. Objaśnienia w tekście.

Jednak praktyka wykazała, że ciągłe rozbudowywanie stosunkowo prostego na początku języka GML prowadzi do stopniowo rosnących trudności z jego implementacjami w konkretnych systemach programowych. Do chwili obecnej żadne oprogramowanie geoinformacyjne obsługujące ten język nie ma zaimplementowanej pełnej wersji GML 3.2.1, a te, które mają ją zaimplementowaną częściowo również mają problemy z interoperacyjnością. Zapisy generowane przez jedne systemy nie są poprawnie przyjmowane przez inne. Dążenie do jak najdalej posuniętej uniwersalności i elastyczności rodzi także wiele problemów. Przykładem są nieskończone wzajemne zagnieżdżenia elementów: „krzywa" `(Curve)` – jako typ geometrii jest podtypem „krzywej abstrakcyjnej" `AbstractCurve)`, która to może mieć także podtyp „krzywa złożona" `(CompositeCurve)`, który z kolei składa się z elementów „składniki krzywej" `(curveMember)` zawierajce typ „krzywa abstrakcyjna" `(AbstractCurve)`, która jak wcześniej może mieć podtyp „krzywa złożona" `(CompositeCurve)` i tak w nieskończoność. Inny przykład nieskończonego zagnieżdżenia przedstawia rysunek 2.16. W tym przypadku element `_FeatureCollection` może zawierać elementy typu `FeatureMember`, których składnikami mogą być ponownie elementy typu `_FeatureCollection`, co w sposób oczywisty prowadzi do nieskończonego zagnieżdżenia. W wielu przypadkach nie powoduje to poważnych błędów zapisu, jednak stwarza realne zagrożenie zapętlenia się procedury kodowania zbioru danych lub jego interpretacji. Wymownym przykładem jest generowanie plików testowych zapisu danych GML na podstawie rozpatrywanego pliku XSD definiującego elementy schematu aplikacyjnego GML.

Problemy implementacyjne języka GML stały się przyczyną coraz częstszych opinii krytycznych. Głębsze zapoznanie się z tymi głosami krytycznymi wskazuje, że są uzasadnione. Zespół roboczy OGC zajmujący się rozwojem języka GML zainicjował otwartą dyskusję nad podstawowymi problemami, jakie powinny być rozwiązane w nowej wersji tego języka, która będzie oznaczona jako 4.0 (Burggraf, 2011). Równolegle do tych działań prowadzone

były prace nad uzupełnieniem obecnej oficjalnej wersji 3.2.1 w zakresie wcześniej zgłoszonych uwag. Na początku roku 2012 została opublikowana nowa wersja 3.3. Jest to jednak tylko rozszerzenie wersji poprzedniej i nie zmienia istotnie funkcjonalności całego języka:

- wprowadzono dwa nowe typy proste: `LanguageString` i `TimePositionUnion`;
- elementy geometryczne zostały rozszerzone o typy uproszczone: `SimplePolygon`, `SimpleRectangle`, `SimpleTriangle`, `SimpleArcString`, `SimpleArc`, `SimpleArcByCenterPoint`, `SimpleArcStringByBulge`, `SimpleArcByBulge`, `SimpleCircle`, `SimpleCircleByCenterPoint`, `SimpleMultiPointType` i `MultiPointProperty`;
- dodano schemat aplikacyjny dla siatek trójkątnych (TIN), w tym elementy: `TriangulatedSurface`, `SimpleTrianglePatch`, `TIN`, `TINElement`, `TINElementProperty` i `TINElementType`;
- dodano schemat dla układów odniesień liniowych z elementami: `PositionExpression` z `PositionExpressionProperty`, `LinearElement` z `LinearElementProperty`, `StartValue`, `LinearReferencingMethod` z `LinearReferencingMethodProperty`, `DistanceExpression` z `DistanceExpressionProperty`, `AlongReferent` z `AlongReferentProperty`, `Referent` z `ReferentProperty`, `Measure`, `LRMName` z `LRMType`, `ReferentType`, `LinearSRS` z `LinearSRSProperty`, `DualAlongReferent` z `DualAlongReferentProperty`, `LRMWithOffset` z `LRMWithOffsetProperty`, `LateralOffsetDistanceExpression`, `VerticalOffsetExpression`, `LateralOffsetDirection`, `VerticalOffsetDirection`, `LateralOffsetLinearSRS` z `LateralOffsetLinearSRSProperty`, `VectorOffsetDistanceExpression`, `VectorOffsetExpression`, `VectorOffsetLinearSRS`;
- uzupełniono pakiet dla pokryć o schematy dla nietypowych siatek, dla których może być określone odniesienie przestrzenne, w tym elementy: `AbstractReferenceableGrid` z `ReferenceableGridProperty`, `ReferenceableGridByArray`, `ReferenceableGridByVectors`, `ReferenceableGridByTransformation` i `gridCRS` z `GridCRSProperty`.

Zasadnicze zmiany w języku GML są zapowiadane w wersji 4.0. Będą dotyczyły głównie modularyzacji całości tak, aby można było dla konkretnych zastosowań wybrać tylko potrzebne moduły, a nie jak jest to obecnie, że w każdym przypadku muszą być zastosowane wszystkie schematy GML, ponieważ ich wzajemne powiązanie tego wymaga. Proponowany podział na moduły został przedstawiony następująco: *gmlCoreProfile, gmlMeasuresProfile, gml2dFeatureProfile, gml3dFeatureProfile, gmlDictionaryProfile, gmlCRSProfile, gmlGridCoverageProfile, gmlAllCoverageProfile, gmlObservationsAndMeasurementsProfile, gmlTemporalProfile* i *gmlDynamicFeatureProfile*.

Druga istotna zmiana w koncepcji organizacji języka GML to opracowanie wielu profili (zawężeń) mających za zadanie także znaczne ograniczenie objętości schematów i liczby elementów wykorzystywanych w różnych zastosowaniach praktycznych. Proponowane są następujące kategorie profili:

- trzy grupy profili dla wyróżnień *(features) – Profile Schemas for Restricted Feature Data* (dla danych o ograniczonym dostępie), *Profile Schemas for NSDI Foundation Data* (dla podstawowych danych w infrastrukturach) i *Profile Schemas for NGA Product* (dla produktów służb geodezyjnych);

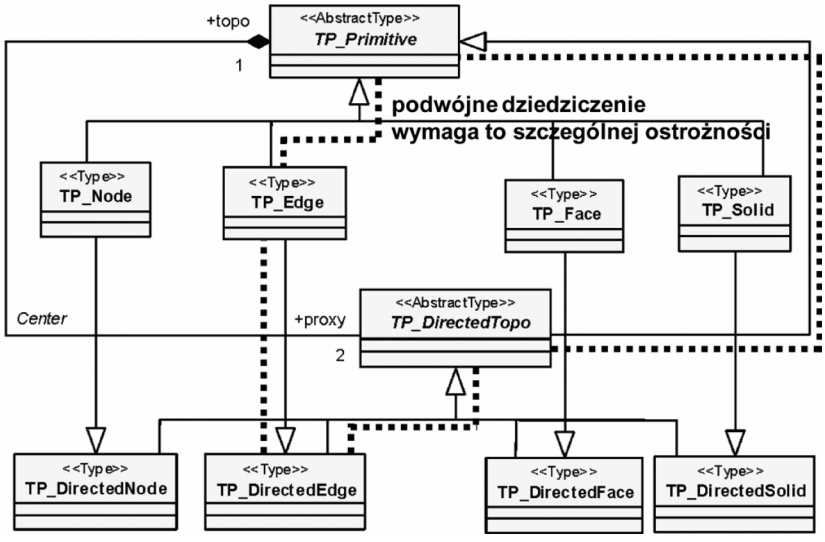
- dwie grupy profili dla geometrii – *Profile Schemas for Geometries* (geometrie bez topologii) i *Profile Schemas for Topology and Geometry* (geometrie z topologią);
- dwie grupy profili dla pokryć *(coverages) – Profile Schemas for Grid Coverages* (włącznie z ortoobrazami) i *Profile Schemas for Non-Grid Coverages* (dla pokryć innych niż siatkowe);
- dwie grupy profili dla układów odniesienia i transformacji danych między układami: *Profile Schemas for Coordinate Reference Systems* (w zakresie definicji układów) i *Profile Schemas for Coordinate Operations* (w zakresie transformacji współrzędnych);
- grupa profili dla usług sieciowych – *Profile Schemas for Specific OGC Services*;
- grupa profili w zakresie metadanych – *Profile Schemas for ISO19139*.

Można oczekiwać, że planowane zmiany organizacyjne w GML 4.0 wyeliminują obecną sytuację, w której musimy się posługiwać wielkim i ciężkim monolitem w każdym zastosowaniu, nawet najprostszym.

2.6. Modele UML dedykowane zapisom w języku GML

Tematem monografii jest transformacja modeli danych zapisanych w UML do form przyjętych w GML i bazach danych. Zakłada się, że takie modele już istnieją i są opracowane poprawnie pod względem merytorycznym i formalnym, a także spełniają wymagania niezbędne do wykonania takiej transformacji. Bardzo często jednak tak nie jest i więcej szczegółowych problemów z tego zakresu jest przedstawione w rozdziałach 9 i 11. Z tego względu trzeba przedstawić podstawowe zasadny opracowywania modelu danych w UML i szerszy kontekst technologii modelowania danych geoprzestrzennych.

W roku 1998 w głównych centrach światowych zajmujących się geoinformacją, w Open Geospatial Consortium (wtedy jeszcze Open GIS Consortium) i w komitecie technicznym ISO/TC 211 podjęto decyzję o zastosowaniu języka UML do opisu modeli danych. Wkrótce



Rys. 2.17. Przykład podwójnego dziedziczenia, które nie może być zaimplementowane w schemacie aplikacyjnym XSD języka GML.
Przykład pochodzi z normy ISO 19108 (ISO/TC 211, 2002b) i dotyczy topologii czasu.

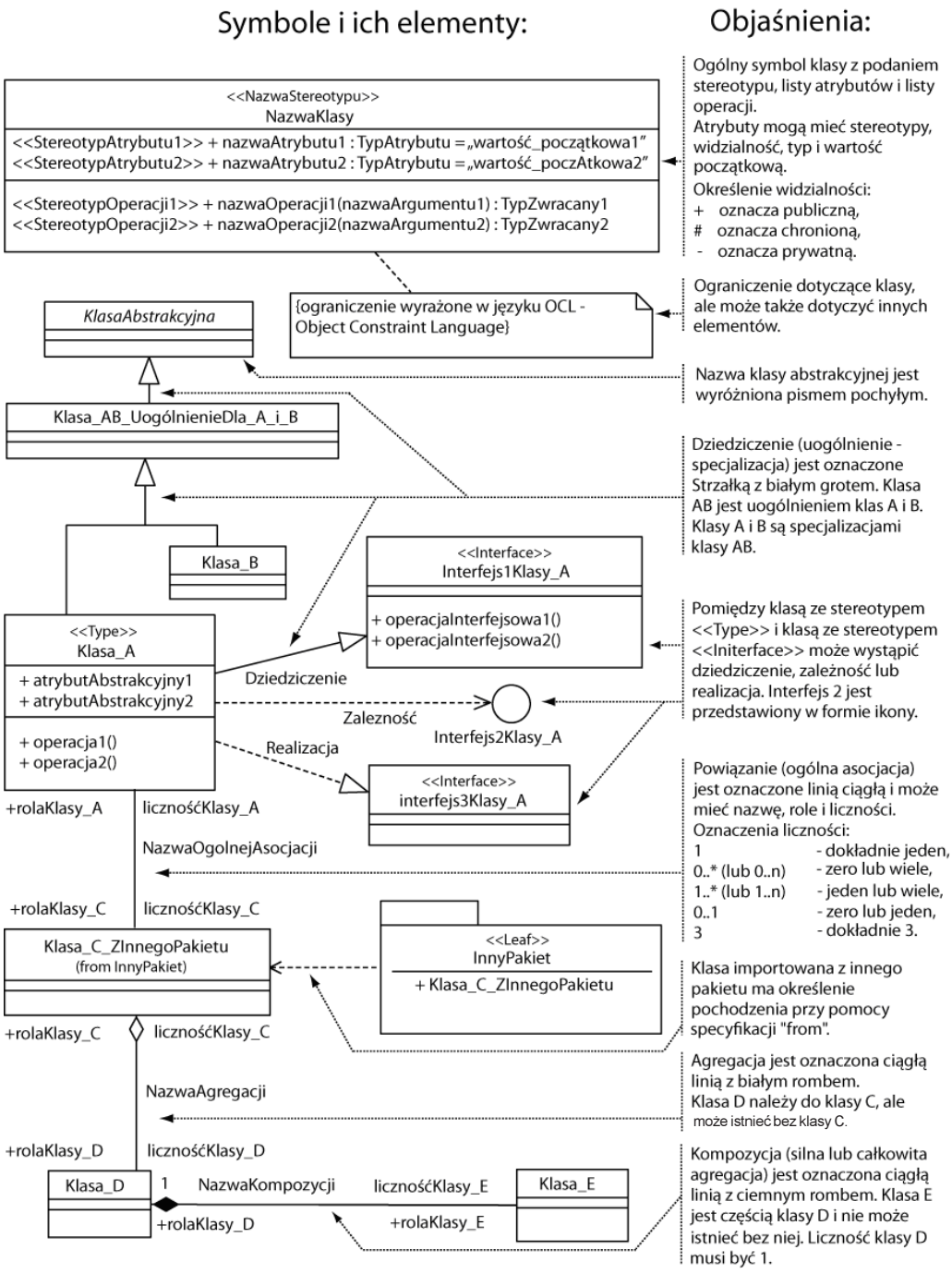
zdefiniowano profil tego języka do geoinformacji i od tej pory wszystkie specyfikacje, standardy i normy posługują się tym językiem do okreslania struktur danych przestrzennych. W tym czasie jeszcze nie myślano o powszechnym zastosowaniu zapisu danych w postaci znacznikowej na bazie XML – pierwsze prace nad tym zainicjowano w OGC w roku 1999, co bardziej szczegółowo opisano w rozdziale 2.3. Długo jeszcze w latach następnych powstawały tam modele nie uwzględniające wymagań transformacji do języka GML. Wiele oficjalnych dokumentów standaryzacyjnych OGC i norm ISO, które powstały w tamtym okresie i są nadal stosowane posługuje się profilem UML zawierającym konstrukcje niemożliwe do przeniesienia do języka GML. Do tej kategorii należy wielokrotne dziedziczenie, którego przykład przedstawia rysunek 2.17.

Podstawowe elementy języka UML stosowane w opisie modeli danych geoprzestrzennych są wyszczególnione na rysunku 2.18. Jednak część przedstawionych elementów i konstrukcji nie może lub nie powinna być stosowana w przypadku, gdy model ma być transformowany do schematu XSD dla języka GML. Należą do nich:

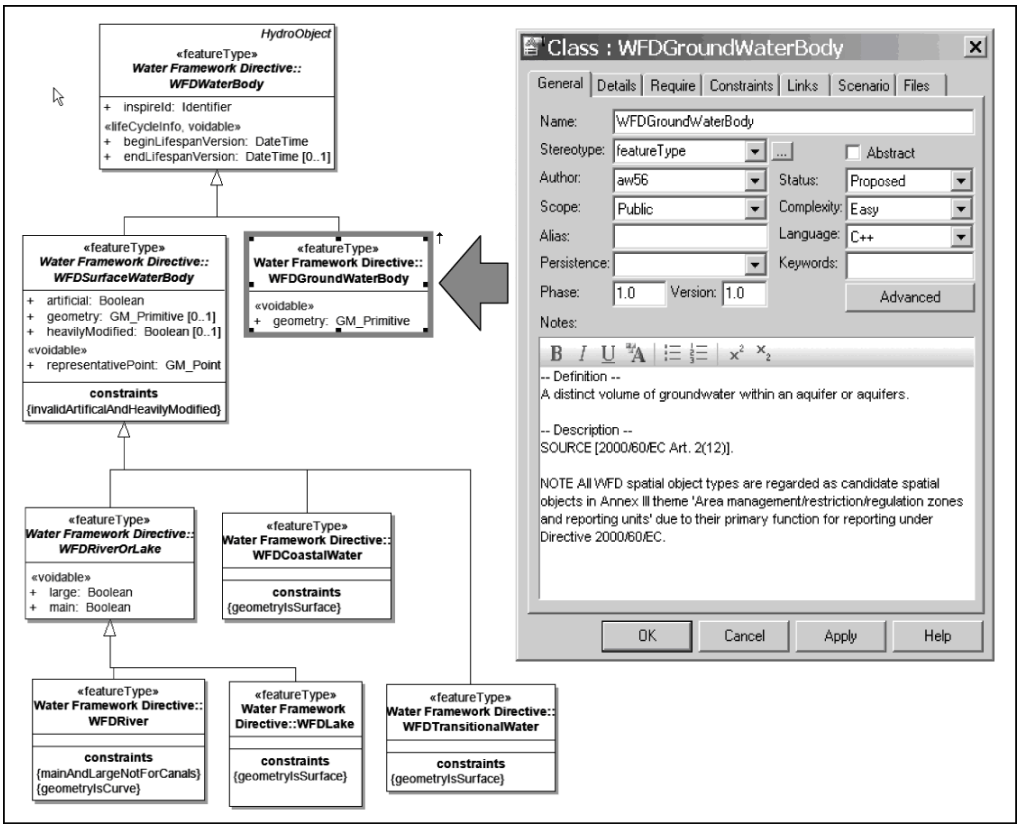
- operacje wyszczególnione w trzeciej od góry sekcji graficznego symbolu klasy (poniżej sekcji atrybutów), ponieważ nie mają znaczenia w opisie struktur danych,
- ograniczenia *(constrains)* zarówno w OCL *(Object Constrain Language),* jak i w innej formie, ponieważ nie są transformowalne do GML,
- podwójne (i wielokrotne) dziedziczenia nie mają swojego odpowiednika w aplikacjach XML,
- wszelkiego typu klasy interfejsowe i asocjacyjne również nie mogą być odwzorowane w GML,
- powiązania pomiędzy klasami typu realizacja lub zależność nie mają odpowiednika w XML,
- stosowanie agregacji stwarza trudności implementacyjne i praktycznie we wszystkich przypadkach może być zastąpione asocjacją jedno- lub dwukierunkową,
- kompozycje mogą być również zastępowane asocjacjami, jednak w takim przypadku nie może być zagwarantowana konieczność usuwania elementów składowych, gdy zostaje ze zbioru danych usunięty element główny – w takim przypadku można to zastąpić atrybutem typu klasa składowa w klasie głównej.

Język UML jest określany jako graficzny język do obrazowania, specyfikowania tworzenia i dokumentowania elementów systemów informatycznych (Booch, Rumbaugh i Jakobson, 2002). W opisywanej problematyce zastosowanie języka UML jest ograniczone jedynie do modeli danych. Trzeba wyraźnie podkreślić, że język UML nie jest jedynie notacją graficzną. Graficzne diagramy tego języka przedstawiają tylko najważniejsze elementy modelu (część lewa rysunku 2.19), a wiele szczegółów modelu pojęciowego nie jest na diagramach bezpośrednio widoczna. Aby je przeczytać lub edytować oprogramowanie narzędziowe języka UML pozwala na otworzenie dodatkowego okna tekstowego (część prawa rysunku 2.19).

W zastosowaniach dotyczących geoinformacji wprowadzono szereg reguł, miedzy innymi w zakresie stosowania nazw w modelach i schematach XSD. Zasady te są istotne, szczególnie w przypadku infrastruktury INSPIRE, gdzie występuje problem wielojęzyczności, zarówno w aspekcie danych, jak i w aspekcie usług sieciowych związanych z tymi danymi. W myśl tych zasad nazwy klas danych, typów danych, atrybutów i elementów występujących w zapisach w języku UML, XML i jego aplikacjach są słowami kluczowymi w sensie informatycznym. Z tego względu w każdym przypadku zachowuje się ich dokładną pisownię w języku angielskim. Nazwy wielowyrazowe pisane są tak zwanym „wielbłądem" *(CamelCase)*, to znaczy nazwy wielowyrazowe są pisane bez przerw (spacji) i poszczególne



Rys. 2.18. Notacja graficzna diagramów klas UML w zakresie profilu ISO dla modeli pojęciowych danych geoprzestrzennych (Michalak, 2003a).



Rys. 2.19. Przykład modelu danych geoprzestrzennych w UML. Po lewej – przykład diagramu klas modelu danych geoprzestrzennych. Po prawej – okno właściwości elementu modelu przedstawiające jedynie część elementów niewidocznych na diagramie (Michalak i inni, 2011).

wyrazy zaczynają się od dużej litery. Nazwy klas, typów i elementów rozpoczynają się dużą literą, a pozostałe małą na przykład: `ToJestNazwaKlasy`, a `toJestNazwaAtrybutu`. Często dla lepszej czytelności akronimy występujące w nazwach oddzielane są podkreśleniem *(underscore)*, na przykład `CGI_Term` – termin z zakresu nazw stosowanych przez CGI *(Commission for the Management and Application of Geoscience Information)*.

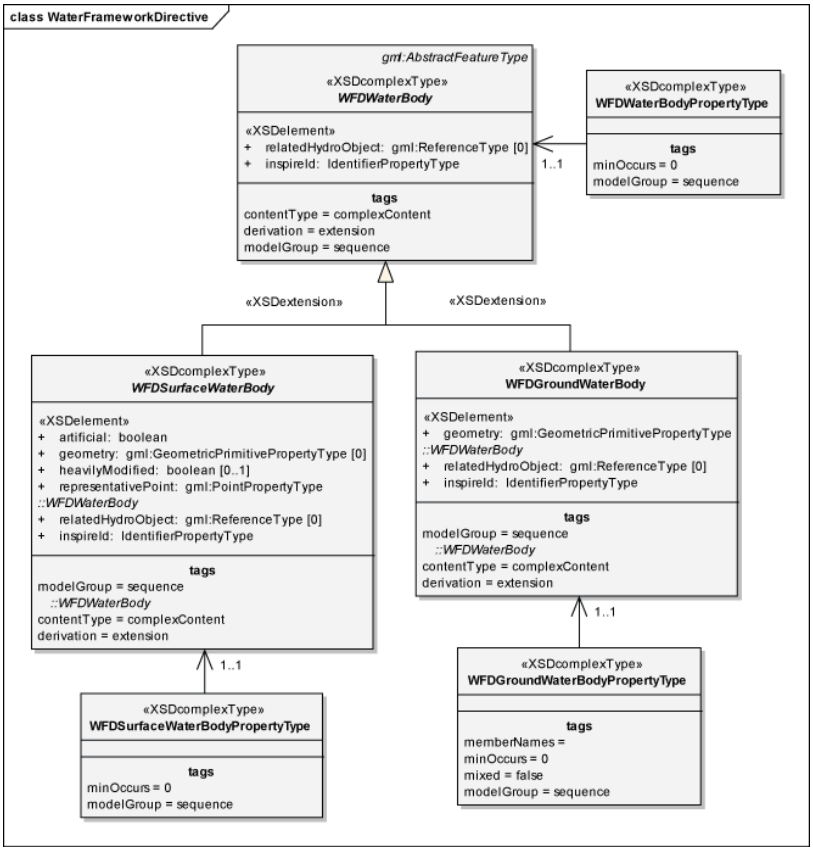
Modele danych, w tym także geoprzestrzennych, są najczęściej opracowywane na dwóch poziomach ogólności:

- modele pojęciowe niezależne od platformy implementacyjnej określane akronimem PIM *(Platform-Independent Model),* często także nazywane modelami abstrakcyjnymi, na przykład w specyfikacjach abstrakcyjnych OGC,
- modele dedykowane określonym platformom (PSM – *Platform-Specific Model*), na przykład zapisom znacznikowym w językach XML i GML, językom programowania C++ lub Java albo relacyjnym lub obiektowym bazom danych.

Więcej na temat tych dwóch kategorii modeli w ujęciu metodyki MDA *(Model-Driven Architecture)* zawierają rozdziały 3 i 10. Tu warto jeszcze zwrócić uwagę na różne podejścia do transformacji modelu abstrakcyjnego (PIM) do modelu dedykowanego określonej platfor-

mie (PSM). Aby model niezależny (PIM) przekształcić w model zależny od platformy (PSM) trzeba w nim dokonać zmian. Konieczność tych zmian wynika z ograniczeń, jakie narzuca ta platforma. Inne ograniczenia są niezbędne w przypadku relacyjnej bazy danych, a inne w przypadku znacznikowego zapisu danych na bazie XML. W rezultacie oba modele wynikowe mogą się znacznie różnić, chociaż zamiar był taki, aby były do siebie jak najbardziej podobne. Często transformacja modelu PIM do modelu PSM jest tak głęboka, że proces ten należy traktować jako nieodwracalny, ponieważ pewne elementy modelu lub jego konstrukcje są tracone, a ponadto podczas transformacji powstają nowe elementy i konstrukcje. Dobitnym tego przykładem jest język GML. Specyfikacja tego języka (Portele, 2007 lub ISO/TC 211, 2007a) zawiera wskazówki i reguły transformacji wprost (UML do GML – aneks E) i odwrotnej (GML do UML – aneks F), jednak transformacja odwrotna (nazywana także inżynierią odwrotną) jest tak trudna, że nie są znane przypadki jej zastosowania. Rysunek 2.20 przedstawia wynik transformacji odwrotnej wykonanej w sposób zgodny z ogólnie przyjętymi regułami w metodyce opartej na języku UML. Uzyskany tą drogą wynik jest zupełnie różny, niż pierwotna źródłowa postać modelu danych, na której podstawie uzyskano schemat aplikacyjny XSD języka GML.

W rozdziale tym przedstawionych jest tylko kilka wybranych najważniejszych problemów dotyczących wymagań języka GML w stosunku do modeli danych w języku UML. Inne przykłady są opisane w rozdziałach 9 i 11.



Rys. 2.20. Przykład zastosowania inżynierii odwrotnej – odtworzenie modelu danych UML na podstawie schematu XSD języka GML. W przykładzie wykorzystano jeden ze schematów tematu INSPIRE: Gospodarowanie obszarem, strefy ograniczone i regulacyjne oraz jednostki sprawozdawcze *(Area management/ restriction/ regulation zones and reporting units)*.

Problematyka niniejszej monografii stanowi przedmiot szerokiego zainteresowania środowisk współtworzących i współużytkujących infrastrukturę informacji przestrzennej budowaną w Polsce zgodnie z przepisami krajowymi i unijnymi. Zainteresowanie to znalazło swój wyraz w warsztatach „Modele danych przestrzennych w UML i ich transformacja do schematów GML i struktur baz danych", które odbyły się w ramach konferencji Polskiego Towarzystwa Informacji Przestrzennej na temat „Informacja przestrzenna dla Polski i Europy", Warszawa, 7–9 listopada 2011 roku. Odpowiadając na ujawnione wówczas zapotrzebowanie, zespół wykładowców podjął trud zawarcia zaprezentowanych przez siebie treści w opracowaniu o charakterze monograficznym. W rezultacie powstała publikacja, która przedstawia w sposób uporządkowany bogaty zasób wiadomości określonych tytułem warsztatów i dotyczących wybranych metod i technologii geoprzestrzennych.

Godne uznania jest, że zespół autorski w składzie: dr inż. A. Chojka, dr inż. A. Zwirowicz-Rutkowska, dr inż. Z. Parzyński i dr hab. J. Michalak, pełniący rolę redaktora naukowego, zrealizował podjęte przedsięwzięcie w stosunkowo krótkim terminie z niewątpliwą korzyścią dla potencjalnych Czytelników.

Jerzy Gaździcki

Warszawa, maj 2012 r.

Autorzy

dr hab. Janusz Michalak
Wydział Geologii, Uniwersytet Warszawski
J.Michalak@uw.edu.pl
Redakcja naukowa i rozdziały:
1. Wstęp
2. Różnice pomiędzy językiem zapisu danych i jego dziedzinową aplikacją
9. Najczęściej popełniane błędy w modelach UML dla schematów aplikacyjnych GML
11. Schematy aplikacyjne tematów aneksów II i III Dyrektywy INSPIRE
12. Podsumowanie

Słownik podstawowych terminów stosowanych w tekście

dr inż. Agnieszka Chojka
Wydział Geodezji i Gospodarki Przestrzennej, Uniwersytet Warmińsko-Mazurski
agnieszka.chojka@uwm.edu.pl
Rozdziały:
3. Wprowadzenie do modelowania informacji przestrzennej – metodyka MDA i diagramy klas UML
6. Budowa schematu aplikacyjnego GML – reguły budowy, narzędzia i przykłady
7. Transformacja schematu aplikacyjnego UML do schematu aplikacyjnego GML – wymagania, ograniczenia i wybrane narzędzia

dr inż. Agnieszka Zwirowicz-Rutkowska
Wydział Geodezji i Gospodarki Przestrzennej, Uniwersytet Warmińsko-Mazurski
agnieszka.zwirowicz@uwm.edu.pl
Rozdziały:
4. Przegląd standardów i narzędzi wykorzystywanych do modelowania informacji geograficznej
5. Schematy aplikacyjne UML – reguły budowy i przykłady
10. Zastosowanie metodyki MDA – wybrane zagadnienia transformacji schematów aplikacyjnych UML do struktur relacyjnych baz danych

dr inż. Zenon Parzyński
Wydział Geodezji i Kartografii, Politechnika Warszawska
z.parzynski@gik.pw.edu.pl
8. Przykład zastosowania metod modelowania danych z zakresu Służby Geodezyjno-Kartograficznej

POLSKIE TOWARZYSTWO INFORMACJI PRZESTRZENNEJ
ROCZNIKI GEOMATYKI 2012 ❍ Tom X ❍ Zeszyt 1(51)

MODELE DANYCH PRZESTRZENNYCH W UML I ICH TRANSFORMACJA DO SCHEMATÓW GML I STRUKTUR BAZ DANYCH

Słowa kluczowe: geoinformacja, informacja geograficzna, model pojęciowy, UML, schemat aplikacyjny, GML, model relacyjny, transformacja

Streszczenie

Celem monografii jest przedstawienie czytelnikom podstawowych metodyk, technik i narzędzi przeznaczonych do budowy modeli pojęciowych danych przestrzennych na poziomie pojęciowym i implementacyjnym, a następnie do transformacji tych modeli do schematów XSD bazujących na języku GML i do zapisów struktur baz danych w języku DDL. Całość składa się z dwunastu rozdziałów dotyczących poszczególnych aspektów budowy modeli i ich transformacji. Wstęp wprowadza czytelników w całą przedstawianą problematykę i naświetla szerszy teoretyczny kontekst z zakresu modelowania i wykorzystania modeli w zastosowaniach praktycznych. Rozdział drugi poświęcony jest nowym metodom zapisu danych przestrzennych opartego na językach znacznikowych, a w szczególności na języku GML, objaśnia zasady takiego zapisu, zawiera krótką historię języka GML i przedstawia inne języki znacznikowe z nim powiązane. Rozdziały trzeci i czwarty stanowią wprowadzenie do modelowania informacji przestrzennej opartego o metodykę MDA z wykorzystaniem języka UML i zawierają przegląd standardów i narzędzi służących temu modelowaniu. W rozdziałach piątym i szóstym przedstawione są podstawowe zasady budowy tematycznych schematów aplikacyjnych w metodyce języka UML i języka GML zilustrowane przykładami. Rozdział siódmy poświęcony jest zagadnieniom transformacji schematów aplikacyjnych z UML do GML, a w szczególności wymaganiom i ograniczeniom, jakie muszą być spełnione, a także dostępnym narzędziom. Kolejny ósmy rozdział dotyczy modeli UML dedykowanych komponentowi infrastruktury krajowej, przeznaczonym dla Służby Geodezyjnej i Kartograficznej. W rozdziale dziewiątym dokonany jest przegląd najczęściej popełnianych błędów w budowie modeli UML przeznaczonych do utworzenia schematów bazujących na języku GML. Tematem rozdziału dziesiątego jest zastosowanie metodyki MDA do transformacji modeli UML do struktur relacyjnych baz danych. Rozdział jedenasty zawiera metodyczną analizę różnych przypadków występujących w modelach danych tematów aneksów II i III dyrektywy INSPIRE, w tym porównanie z modelami tematów aneksu I, analizę różnych typów i form danych, jakie tam występują. Dwunasty rozdział to podsumowanie, w którym zwraca się szczególną uwagę na dynamiczny rozwój metod z tego zakresu, zmiany zachodzące w zakresie stosowanej terminologii i skutki, jakie te zmiany za sobą pociągają.

UML GEOSPATIAL DATA MODELS AND THEIR TRANSFORMATION INTO GML SCHEMAS AND DATABASE STRUCTURES

Keywords: geoinformation, geographic information, conceptual model, UML, application schema, GML, relational model, transformation

Abstract

The main objective of the monograph is to present essential methodologies, technologies and software tools dedicated to building conceptual models of geospatial data on conceptual level, and implementation level, and then to be transformed into XSD schemas based on GML language and to encode data bases structures in DDL language. The whole monograph consists of twelve chapters concerning different aspects of models development and their transformation. The introduction familiarizes readers with all issues presented and clarifies broader theoretical context with regard to modeling and exploitation of models in practical applications. The second chapter is dedicated to modern methods of encoding spatial data based on markup languages, in particular on GML language; rules for that encoding are also explained. This chapter contains a short history of GML language and presents other markup languages associated with it. Chapters three and four provide an introduction to spatial information modeling based on MDA methodology with application of UML language and it contains a review of standards and tools dedicated to such modeling. In chapters five and six, essential rules of development of thematic application schemas are presented in the methodology of UML and GML languages. Examples to illustrate them are provided. Chapter seven is dedicated to issues of transformation application schemas from UML to GML, in particular to the requirements and constrains that must be fulfilled and also to available tools. The next chapter eight concerns UML models dedicated to components of the national infrastructure designated for Geodetic and Cartographic Service. In chapter nine, a review of most frequent mistakes committed in drawing up UML models dedicated to generating of schemas based on GML language are presented. The subject of chapter ten is the application of MDA methodology for transformation of UML models into relational databases structures. Chapter eleven contains methodological analysis of various cases occurring in data models for the themes defined in Annex II and III of INSPIRE Directive as well as a comparison with the models for themes defined in Annex I and an analysis of various data forms occuring there. In chapter twelve, the recapitulation is presented, in which dynamic development of methods in this area is taken in consideration. In addition, significant changes in the terminology and the effects of these changes are discussed.

POLSKIE TOWARZYSTWO INFORMACJI PRZESTRZENNEJ
ROCZNIKI GEOMATYKI 2012 ❍ Tom X ❍ Zeszyt 1(51)

Spis treści

POLSKIE TOWARZYSTWO INFORMACJI PRZESTRZENNEJ
ROCZNIKI GEOMATYKI 2012 ❍ Tom X ❍ Zeszyt 1(51)

Janusz Michalak

Słownik podstawowych terminów stosowanych w tekście

Abstrakcyjny – obiekt, atrybut, typ, klasa *(abstract – object, attribute, type, class)* – **1:** Określony ogólnie, bez szczegółów związanych z określoną implementacją (uwarunkowaniami technologicznymi) lub z określoną aplikacją (uwarunkowaniami wynikającymi z dziedziny zastosowania). Na przykład wynik pomiaru w znaczeniu ogólnym jako atrybut w modelu pojęciowym nie musi mieć określonego typu. Typ będzie zależał od fizycznego charakteru mierzonego elementu i od typy przyrządu pomiarowego. **2:** Klasa abstrakcyjna w modelu danych to klasa, która nie ma własnych obiektów, a jedynie służy jako klasa bazowa dla innych klas. Użycie takiej klasy jest uzasadnione tylko gdy są (lub mogą być) wyprowadzone z nie przynajmniej dwie klasy.

Atrybut *(attribute)* – Właściwość wyróżnienia lub obiektu określona przez nazwę tej właściwości i zakres wartości, jakie mogą być przypisane tej nazwie dla określenia tej właściwości.

Atrybut geoprzestrzenny *(geospatial attribute)* – Właściwość (cecha) wynikająca z faktu, że wyróżnienie zajmuje pewne miejsca w rzeczywistości w sensie geoprzestrzennym. Najczęściej przez domniemanie przyjmuje się, że określenie geoprzestrzenny obejmuje również czas, czyli jest równoznaczne z określeniem czaso-geoprzestrzenny. Przykładami takich atrybutów są: wielkość, kształt, położenie, przynależność geoprzestrzenna (np. leży w obrębie), relacje geoprzestrzenne względem innych wyróżnień (np. odległość lub rodzaj sąsiedztwa).

Atrybut niegeoprzestrzenny *(non-geospatial attribute)* – Wszystkie pozostałe atrybuty niezwiązane z odniesieniem przestrzennym. Atrybuty te mogą należeć zarówno do wyróżnień geoprzestrzenne jak i do innych obiektów i wystąpień niegeoprzestrzennych.

Cecha *(trait)* – Kategoria klasy, której zadaniem jest (w przypadku modeli danych) dostarczenie innej klasie określonych własności (atrybutów i powiązań z innymi klasami). W tym przypadku klasa ma stereotyp «`trait`». Podobnym mechanizmem pozwalającym na uniknięcie problemów wielokrotnego dziedziczenia jest **domieszka**.

Dane (*data*, w liczbie pojedynczej: *datum*) – **1:** Jednostki informacji, czyli pojedyncze fragmenty informacji. Dane niezorganizowane nie stanowią informacji i często są bezużyteczne. Dane zorganizowane stanowią elementy informacji. Zorganizowanie danych może być jawne, na przykład w językach znacznikowych lub niejawne, na przykład miejsce umieszczenia adresu na kopercie decyduje, czy jest to adres nadawcy czy odbiorcy. **2:** Fakty, statystyki, opinie i przewidywania zebrane z różnych wewnętrznych i zewnętrznych źródeł. Dane bez kontekstu są szumem (Nowicki i Staniszkis, 2002).

Dane geoprzestrzenne *(geospatial data)* – **1:** Dane w sensie zdefiniowanym przez informatykę, ale w odróżnieniu od innych rodzajów danych są one odniesiona do określonego miejsca (fragmentu przestrzeni) i w rezultacie niezbędnymi ich składnikami są dane określające położenie tego miejsca względem Ziemi. **2:** Dane przestrzenne dotyczące Ziemi i wszystkich obiektów przestrzennych z nią związanych (Gaździcki, 2004).

Domieszka *(mixin)* – **1:** Kategoria klasy, której zadaniem jest (w przypadku modeli danych) dostarczenie innej klasie określonych własności (atrybutów i powiązań z innymi klasami). Taka klasa nie ma własnych obiektów, czyli musi być abstrakcyjna. Stosowanie tego rodzaju klasy jest uzasadnione tyko w przypadkach, gdy przynajmniej dwie zwykłe klasy otrzymują w ten sposób własności. Jest to sposób na uniknięcie problemów z implementacją wielokrotnego dziedziczenia. Jedyny przypadek zastosowania klasy *mixin* do języka GML to modele dla niektórych tematów INSPIRE. **2:** Ograniczony sposób dziedziczenia ma pozwalający również na uniknięcie problemów z implementacją wielokrotnego dziedziczenia. W takim przypadku powiązanie dziedziczenia ma stereotyp «mixin». Porównaj: **cecha**.

Encja *(entity)* – Pojęcie z modelu encja-związek, oznaczające konkretny lub abstrakcyjny byt wyróżnialny w modelowanej rzeczywistości. W odróżnieniu od obiektu, encja nie jest kojarzona z metodami (Subieta, 1999a).

GML *(Geography Markup Language)* – Język znaczników geograficznych, aplikacja języka (metajęzyka) XML przeznaczona do zapisu geoinformacji w celu przesyłania jej pomiędzy różnymi systemami – on-line, niezależnie od platformy sprzętowo-systemowej i niezależnie od charakteru i technologii systemu geoinformacyjnego (Gaździcki, 2004).

Informacja *(information)* – **1:** Dane komputerowe, które są zorganizowane i przedstawione w usystematyzowanej formie dla zrozumiałości ich podstawowego znaczenia. Związki pomiędzy informacją i danymi wyjaśnia definicja danych. **2:** Dane interpretowane w kontekście określonego celu (Nowicki i Staniszkis, 2002). **3:** Wiedza uzyskiwana w drodze interpretacji danych, która w ustalonym kontekście ma określone znaczenie i dotyczy obiektów, takich jak fakty, zdarzenia, przedmioty, zjawiska, procesy i idee (Gaździcki, 2004).

Informacja geograficzna – patrz: **informacja geoprzestrzenna**.

Informacja geoprzestrzenna *(geospatial information)* – **1:** Informacja w sensie zdefiniowanym przez informatykę, ale w odróżnieniu od innych rodzajów informacji jest ona odniesiona do określonego miejsca (fragmentu przestrzeni) i w rezultacie niezbędnymi jej składnikami są dane określające położenie tego miejsca względem Ziemi. **2:** Informacja uzyskiwana w drodze interpretacji danych geoprzestrzennych (Gaździcki, 2004).

Instancja *(instance)* – Synonim egzemplarza stosowany w normach PN-EN ISO 19100 (Gaździcki, 2011).

Klasa *(class)* – Pojęcie klasy jest używane w trzech dość bliskich znaczeniach: **(1)** zbiór obiektów o zbliżonych własnościach; **(2)** byt semantyczny rozumiany, jako miejsce przechowywania takich cech grupy podobnych obiektów, które są dla nich niezmienne (np. zestawu atrybutów, nazwy, metod, ograniczeń dostępu); **(3)** wyrażenie językowe specyfikujące budowę obiektów, dozwolone operacje na obiektach, ograniczenia dostępu, wyjątki, itd. Zwykle klasy wiąże się ze sobą poprzez hierarchię (lub inną strukturę) dziedziczenia (Subieta, 1999a).

MDA – 1: *(Model Driven Approach)* Podejście oparte na modelu: pojęciowym, logicznym i fizycznym. Niezależny od implementacji schemat aplikacyjny zostaje odwzorowany na różne specyfikacje (wykorzystujące różne technologie, np. usługi sieciowe, relacyjne bazy danych, XML), a te z kolei mogą zostać zaimplementowane (wdrożone) na różnych platformach sprzętowo-programowych (CEN, 2006). **2:** *(Model Driven Architecture)* Zbiór metod porządkujących proces tworzenia systemów informatycznych opartych na budowie modeli i ich transformacji. Koncepcja MDA została opracowana przez międzynarodową organizację OMG, której celem jest rozwiązywanie problemów związanych z integracją systemów informatycznych pochodzących od różnych dostawców oraz działających na różnych platformach informatycznych (OMG, 2003).

Metamodel *(metamodel)* – W założeniu, model definiujący składnię, semantykę i pragmatykę wprowadzonego modelu, notacji lub diagramu. Metamodel proponowany przez autorów UML ustala pewne elementy składni diagramów, ograniczenia typologiczne, klasyfikację pojęć oraz związki pomiędzy pojęciami (Subieta, 1999a).

Metka *(tagged value)* – Inaczej wartość etykietowana. Obok stereotypów i ograniczeń, to jeden z mechanizmów rozszerzenia semantyki języka UML. Pozwala dołączyć do elementu modelu UML dodatkowe właściwości. Metka to para *klucz=wartość*.

Metodyka *(methodology)* – Zestaw pojęć, notacji, modeli formalnych, języków i sposobów postępowania służący do analizy rzeczywistości (stanowiącej przedmiot projektowanego systemu informatycznego) oraz do projektowania pojęciowego, logicznego i/lub fizycznego. Zwykle metodyka jest powiązana z odpowiednią notacją (diagramami) służącymi do zapisywania wyniku poszczególnych faz projektu, jako środek wspomagający ludzką pamięć i wyobraźnię i jako środek komunikacji w zespołach oraz pomiędzy projektantami i klientem (Subieta, 1999a).

Model pojęciowy *(conceptual model)* – Model procesów lub model struktury danych odwołujący się do ludzkiej percepcji i wyobraźni, mający za zadanie zrozumienie problemu, udokumentowanie wyniku analizy lub projektu w czytelnej i abstrakcyjnej formie językowej oraz ułatwienie komunikacji w zespołach ludzkich (Subieta, 1999a).

Model semantyczny *(semantic model)* – Zestaw pojęć, technik i notacji mający na celu odwzorowanie semantyki danych, czyli ich znaczenia w świecie zewnętrznym. Modele semantyczne wprowadzają w tym celu pojęcia, takie jak: generalizacja, specjalizacja, asocjacja, agregacja, klasyfikacja, własności temporalne, zdarzenia, własności behawioralne, itd. Przykładem prostego modelu semantycznego jest model encja-związek. Niekiedy terminem "model semantyczny" określa się również konkretny diagram (lub inną formę językowo-graficzną) odwzorowującą rzeczywistość opisywaną przez dane (Subieta, 1999a).

Norma *(standard)* – **1:** Dokument przyjęty na zasadzie konsensu i zatwierdzony przez upoważnioną jednostkę organizacyjną, ustalający – do powszechnego i wielokrotnego stosowania – zasady, wytyczne lub charakterystyki odnoszące się do różnych rodzajów działalności lub ich wyników i zmierzający do uzyskania optymalnego stopnia uporządkowania w określonym zakresie (Ustawa, 2002). **2:** Polska Norma – jest normą o zasięgu krajowym, przyjętą w drodze konsensu i zatwierdzoną przez krajową jednostkę normalizacyjną (Polski Komitet Normalizacyjny), powszechnie dostępną, oznaczoną – na zasadzie wyłączności – symbolem PN (PKN, 2010). Zobacz: **normy ISO serii 19100, standard, standardy OGC.**

Normy ISO serii 19100 *(ISO 19100 series of International Standards)* – Rodzina norm ISO w dziedzinie informacji geograficznej. Wynik prac Komitetu Technicznego ISO/TC211, który pracuje nad wieloma projektami standaryzacji informacji przestrzennej w bardzo szerokim zakresie tej problematyki. Zobacz: **norma, standard, standardy OGC**.

Obiekt *(object*) – **1:** W teorii informacji – konkretny lub abstrakcyjny byt (wystąpienie) wyróżnialny w modelowanej rzeczywistości, posiadający nazwę, jednoznaczną identyfikację, wyraźnie określone granice, atrybuty i inne właściwości takie jak rodzaj struktury wewnętrznej lub struktury danych z nim związanych. Te składniki obiektu określają: jego stan (poprzez wartości atrybutów i powiązania) i jego zachowanie się (poprzez operatory i funkcje, czyli metody) (Subieta, 1999a). **2:** W geomatyce przyjmuje się, że obiekt jest wystąpieniem klasy i jest to oparte na paradygmacie obiektowości wywodzącym się z języka UML, który jest przyjęty do opisu modeli pojęciowych (OMG, 2001). **3:** Termin stosowany w różnych znaczeniach; dla uniknięcia wątpliwości, jeśli jego znaczenie nie wynika z kontekstu, powinien być uzupełniony dodatkowym określeniem (Gaździcki, 2004).

Rola *(role)* – W języku UML jedna z możliwości opisu powiązania. Pozostałe to nazwa powiązania oraz krotność. Każda klasa biorąca udział w powiązaniu ogrywa w nim określoną rolę. Inaczej jest to „oblicze", które klasa przy jednym końcu powiązania prezentuje klasie przy drugim jego końcu.

Schemat *(schema)* – **1:** Opis logicznej struktury bazy danych lub innego systemu związanego z danymi, np. interfejsu wymiany danych (XML Schema). **2:** Opis atrybutów wyróżnień *(feature)*, lub bardziej dokładnie – specyficzny model atrybutów dla wyróżnień określony za pomocą elementarnych typów danych i ograniczeń dotyczących tych typów (Buehler, McKee, 1996).

Schemat aplikacyjny *(application schema)* – Schemat przeznaczony dla konkretnego systemu lub dla konkretnej dziedziny zastosowań.

Schemat implementacyjny *(implementation schema)* – Schemat uwzględniający technologiczne środowisko, w którym będzie realizowana jego aplikacja. Na przykład zapisany w formie schematu XML.

Specyfikacja *(specification)* – **1:** Abstrakcyjny opis bytu programistycznego (procedury, modułu, klasy, obiektu, bazy danych, itp.) określający reguły użycia lub ustalający podstawowe założenia jego implementacji (Subieta, 1999a). **2:** Dokument lub opis, który określa w sposób kompletny, precyzyjny i sprawdzalny wymagania, projekt lub charakterystykę systemu lub jego fragmentu, a często także procedury dla określenia czy te wymagania są spełnione.

Standard *(standard)* – Wzorzec rozwiązania sprzętowego lub programowego zatwierdzony przez instytucję normalizacyjną lub przyjęty nieformalnie wskutek dużego upowszechnienia, w przypadku standardów informatycznych najczęściej o zasięgu światowym. Do najważniejszych instytucji opracowujących standardy należą: ISO, IEEE, ANSI. Przykładami standardów są: RS-232-C (fabryczny standard interfejsu sprzętowego), ANSI C++ (oficjalny standard języka programowania), POSIX (standard IEEE przenośnego systemu uniksowego), CORBA (standard obiektowych systemów rozproszonych) (Płoski, 1999). Zobacz: **standardy OGC, norma, normy ISO serii 19100.**

Standardy OGC *(OGC standards)* – Techniczne dokumenty specyfikujące interfejsy i reguły zapisu danych geoprzestrzennych. Stanowią one główne rezultaty działalności OGC *(Open Geospatial Consortium)* i są opracowywane przez zespoły złożone z członków OGC dla rozwiązywania różnorodnych problemów dotyczących interoperacyjności. Wszystkie publiczne dokumenty OGC są łatwo dostępne bez żadnych opłat. OGC ma ponad 400 członków, w tym ponad połowa to wyższe uczelnie i instytucje naukowe, także prawie połowę stanowią członkowie europejscy. Standardy OGC dzielą się na specyfikacje abstrakcyjne i standardy implementacyjne. Wiele z tych standardów zostało przyjęte przez komitet ISO/TC 211 jako normy ISO, na przykład: 19107, 19115, 19119, 19123, 19125, 19128, 19136, 19139, 19142, 19143, i 19156. Ze standardami OGC powiązane są inne oficjalne dokumenty OGC, na przykład: OGC *Reference Model* (ORM), *Engineering Reports* lub nieoficjalne, na przykład *Best Practices Documents* i *Discussion Papers*. Zobacz: **standard, norma, normy ISO serii 19100.**

Stereotyp *(stereotype)* – W terminologii UML, klasyfikacja elementu modelu posiadająca semantyczne konsekwencje. Stereotypy mogą być predefiniowane lub zdefiniowane przez użytkownika (Subieta, 1999a).

Struktura *(structure)* – Termin w C++ (także w innych językach) na oznaczenie zestawu nazwanych wartości, w innych językach odpowiada jej zapis lub rekord (Subieta, 1999a).

Tabela *(table)* – Struktura danych implementowana w relacyjnych bazach danych, często nazywana relacją. Tabela składa się z wierszy lub inaczej krotek. Należy zwrócić uwagę, że pomiędzy relacją (w sensie matematycznym) i tabelą występują dość istotne różnice koncepcyjne. Tabela jest wyposażona w nazwy kolumn (które niosą informację semantyczną) (Subieta, 1999a).

Tożsamość *(identity)* – Tożsamość obiektu oznacza, że obiekt istnieje i jest odróżnialny niezależnie od jego aktualnego stanu (wartości atrybutów), który może się zmieniać; możliwe są dwa różne obiekty o identycznych wartościach atrybutów. Praktycznie, tożsamość oznacza istnienie unikalnego wewnętrznego (nieczytelnego dla użytkownika) identyfikatora obiektu, który nie ulega zmianie podczas życia obiektu (Subieta, 1999a).

Unia *(union)* – Typ struktury, rekordu lub obiektu, który może mieć alternatywnie dwa lub więcej zestawów atrybutów. Przykładowo, jeżeli właścicielem samochodu może być osoba lub firma, to obiekt *Samochód* może posiadać alternatywnie albo atrybut *Nazwisko Właściciela* albo atrybut *WłasnośćFirmy*. Unia może mieć związany dyskryminator *(discriminator)*, tj. atrybut, którego wartość określa, z którym wariantem mamy do czynienia. Może też nie mieć dyskryminatora; wówczas odpowiedzialność za rozróżnianie wariantów spada na programistę (tak jest np. w C i C++). Brak dyskryminatora w unii podkopuje koncepcję mocnej kontroli typów i stwarza okazję do bardzo trudnych błędów (Subieta, 1999a).

Walidator (validator) – Program komputerowy sprawdzający poprawność dokumentu (np. XML) o określonej składni.

Wyróżnienie geoprzestrzenne *(geospatial feature)* (w literaturze polskiej termin *feature* jest często tłumaczony jako obiekt) – **1:** Podstawowy fragment (atom) informacji geoprzestrzennej. Posiada atrybuty geoprzestrzenne (geometryczne i topologiczne) np. kształt, rozciągłość, położenie, relacje z innymi wyróżnieniami. Często pojęcie wyróżnienie jest my-

lone z pojęciem obiekt, jednak wyróżnienie może być obiektem, ale też może nim nie być (Mark i in., 2001). Ponieważ w geomatyce wszystkie wyróżnienia są geoprzestrzenne, przymiotnik geoprzestrzenny jest na ogół pomijany i używa się krótszego terminu wyróżnienie. **2:** Cyfrowa reprezentacja zjawiska (bytu) świata rzeczywistego lub jego abstrakcja w modelu pojęciowym. Wyróżnienie ma określone miejsce w przestrzeni i czasie jako jego atrybuty (Buehler, McKee, 1996). Przykładem wyróżnienia może być prawie wszystko co może być umieszczone w przestrzeni i czasie: stół, budynek, miasto, drzewo, fragment lasu, ekosystem, trasa przejazdu lub wyż atmosferyczny jako obszar wysokiego ciśnienia powietrza. **3:** Abstrakcja zjawiska świata rzeczywistego. Termin wyróżnienie może odnosić się do typu zjawiska lub jego konkretnego wystąpienia (ISO/TC 211, 2002a), np. „rzeka" i „Wisła".

Związek *(relationship)* – **1:** W języku UML i w konsekwencji także w normach grupy ISO 19100 – semantyczne połączenie pomiędzy elementami modelu. Przykładami związków są agregacje, kompozycje (agregacje całkowite), powiązania i uogólnienia. **2:** W modelu encji-relacji – powiązanie pomiędzy encjami (Michalak, 2005a).

Literatura

AB ORMSC (Architecture Board ORMS), 2001: Model Driven architecture (MDA). Document number ormsc/2001-07-01.
URL: http:// http://www.enterprise-architecture.info/Images/MDA/MDA%20Technical.pdf

Altova, XMLSpy. URL: http://www.altova.com/xml-editor

BGWM (Biuro Geodety Województwa Mazowieckiego), 2009: Opis koncepcji identyfikatorów, wersjonowania zmian, stosowania reguły *nil reason*.
URL: http://www.geointegracja.gov.pl/ download/file.php?id=80&sid=f2c1f79e942a2cf12ed12a99aee5eec0

Biron P. V., Permanente K., Malhotra A. (W3C), 2004: XML Schema Part 2: Datatypes. Second Edition. W3C Recommendation 28 October 2004. URL: http://www.w3.org/TR/xmlschema-2

Boisvert E., Brodaric B., 2008: GroundWater Markup Language Specification v. 1.0.
URL: http://ngwd-bdnes.cits.rncan.gc.ca/service/api_ngwds/en/gwml.html

Booch G., Rumbaugh J., Jacobson I., 2002: UML – przewodnik użytkownika. Z serii: Inżynieria oprogramowania. Wydanie polskie. Wydawnictwa Naukowo-Techniczne, Warszawa.

Brink L., Portele C., Vretanos P. A. (OGC), 2011: Geography Markup Language (GML) simple features profile (with Corrigendum). OpenGIS Implementation Standard Profile.
URL: http://portal.opengeospatial.org/files/?artifact_id=42729

Buechler K., McKee L. (ed.), 1996: The OpenGIS Guide – Introduction to Interoperable Geoprocessing – Part I of the Open Geodata Interoperability Specification (OGIS). OGIS TC Document 96-001, Open GIS Consortium, Wayland.

Burggraf D., 2011: Input to the GML 4 workshop.
URL: http://external.opengeospatial.org/twiki_public/GML/Gml4WorkshopInput

CEN, 2006: prCEN/TR 15449, Geographic information – Standards, specifications, technical reports and guidelines, required to implement Spatial Data Infrastructure.

Chojka A., 2006: Przegląd metod, środków formalnych i narzędzi programowych wspomagających modelowanie pojęciowe informacji geograficznej. Część I – Modelowanie pojęciowe. *Magazyn Geoinformacyjny Geodeta*, nr 5 (132).

Cox S. (ed.) (OGC), 2010: Geographic Information: Observations and Measurements – OGC Abstract Specification Topic 20. URL: http://portal.opengeospatial.org/files/?artifact_id=41579

Cox S., 2011: Hollow World: a GML application schema template. Solid Earth and Environment GRID (SEE GRID community website). URL: https://www.seegrid.csiro.au/wiki/AppSchemas/HollowWorld

CTWG-O&M (INSPIRE Cross Thematic Working Group on Observations & Measurements), 2011: D2.9 Guidelines for the use of Observations & Measurements and Sensor Web Enablement-related standards in INSPIRE Annex II and III data specification development, Version 1.0.
URL: http://inspire.jrc.ec.europa.eu/documents/Data_Specifications/D2.9_O&M_Guidelines_V1.0.pdf

DT_DS (INSPIRE Drafting Team „Data Specifications"), 2008: D2.6: Methodology for the development of data specifications, Version 3.0.
URL: http://inspire.jrc.ec.europa.eu/reports/ImplementingRules/DataSpecifications/D2.6_v3.0.pdf

DT_DS (INSPIRE Drafting Team „Data Specifications"), 2010a: D2.5: Generic Conceptual Model, Version 3.3. URL: http://inspire.jrc.ec.europa.eu/documents/Data_Specifications/D2.5_v3_3.pdf

DT_DS (INSPIRE Drafting Team „Data Specifications"), 2010b: D2.7: Guidelines for the encoding of spatial data, Version 3.2. URL: http://inspire.jrc.ec.europa.eu/documents/Data_Specifications/D2.7_v3.2.pdf

EC (European Commission), 2009: Guidance Document No. 22 – Updated Guidance on Implementing the Geographical Information System (GIS) Elements of the EU Water policy. Common Implementation Strategy for the Water Framework Directive (2000/60/EC). Technical Report – 2009 – 028. URL: http://circa.europa.eu/Public/irc/ env/wfd/library?l=/framework_directive/guidance_documents/guidance-no22-_nov08pdf_1/_EN_1.0_&a=d

EP&CEU (European Parliament and Council of the European Union), 2007: Directive 2007/2/EC of the European Parliament and of the Council of 14 March 2007 establishing an Infrastructure for Spatial Information in the European Community (INSPIRE).

Fowler M., Scott K., 2002: UML w kropelce. Oficyna wydawnicza LTP, Warszawa.

Gaździcki J., 2004: Leksykon geomatyczny – Lexicon of geomatics. Polskie Towarzystwo Informacji Przestrzennej, Warszawa.

Gaździcki J., 2011: [W:] (red.) Gaździcki J. Internetowy leksykon geomatyczny. URL: http://www.ptip.org.pl/auto.php?page=Encyclopedia&enc=1

Githaiga J., 2010: Project Overview – FullMoon. Solid Earth and Environment GRID (SEE GRID community website). URL: https://www.seegrid.csiro.au/wiki/Siss/FullMoon

Huang C-H., Chuang T-R., Deng D-P., Lee H-M, 2009: Building GML-native web-based geographic information systems. *Computers&Geosciences*, no 35, 1802-1816. URL: http://www.iis.sinica.edu.tw/papers/trc/8843-F.pdf

IGW-CGI-IUGS (Commission for the Management and Application of Geoscience Information – CGI, Interoperability Working Group – IWG, International Union of Geological Sciences – IUGS), 2008: GeoSciML Cookbook – How To Map Data to GeoSciML, Version 2. URL: http://www.geosciml.org/geosciml/2.0/cookbook/GeoSciML_Data_CookBook_V2.pdf

ISO/TC 211 (Geographic Information/Geomatics), 2002a: ISO 19101: Geographic information – Reference model. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26002

ISO/TC 211 (Geographic Information/Geomatics), 2002b: ISO 19108:2002 Geographic information – Temporal schema. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26013

ISO/TC 211 (Geographic Information/Geomatics), 2003: ISO 19107:2003 – Geographic information – Spatial schema. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=26012

ISO/TC 211 (Geographic Information/Geomatics), 2005a: ISO 19103 Technical Specification, Geographic information – Conceptual schema language. URL: http://www.iso.org/iso/catalogue_detail.htm?csnumber=37800

ISO/TC 211 (Geographic Information/Geomatics), 2005b: ISO 19109:2005 Geographic information – Rules for application schema. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39891

ISO/TC 211 (Geographic Information/Geomatics), 2006: ISO 19110 – Geographic information – Methodology for feature cataloguing. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=39965

ISO/TC 211 (Geographic Information/Geomatics), 2007a: ISO 19136:2007 – Geographic information – Geography Markup Language (GML). URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32554

ISO/TC 211 (Geographic Information/Geomatics), 2007b: ISO 19139 Technical Specification, Geographic information – Metadata – XML schema implementation. URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32557

ISO/TC 211 (Geographic Information/Geomatics), 2009: ISO 19104:2009 Technical Specification, Geographic information – Terminology URL: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=32557

ISO/TC 211 (Geographic Information/Geomatics), 2011: ISO 19118, Geographic information – Encoding. URL: http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=44212

ISO/TC 211 (Geographic Information/Geomatics), 2012: Introduction:Welcome to the ISO/TC 211 Harmonized Model Web server. URL: http://www.isotc211.org/hmmg/HTML/root.html

Lake R., Burggraf D., Trninic M., Rae L., 2004: Geography Markup Language: Foundation for the Geo-Web. Wiley (w znacznej części dostępna bezpłatnie)
URL: http://media.wiley.com/product_ancillary/47/04708715/DOWNLOAD/Lake.zip

Mark D. M., Skupin A., Smith B., 2001: Features, Objects, and other Things: Ontological Distinctions in the Geographic Domain. Spatial Information Theory, Proceedings of COSIT 2001, Springer.
URL: http://wings.buffalo.edu/philosophy/faculty/smith/articles/COSIT01MSS.pdf

Michalak J., 2003a: Modele pojęciowe hydrogeologicznych danych geoprzestrzennych – podstawy metodyczne. Biuletyn PIG – *Hydrogeologia*, z. V, nr 406, monografia.

Michalak J., 2003b: Podstawy metodyczne i technologiczne infrastruktur geoinformacyjnych. *Roczniki Geomatyki*, t. 1, z. 2. PTIP, Warszawa, monografia.

Michalak J., 2003c: Geomatics in hydrogeology. *Geological Quarterly*, 47(1): 69-76.

Michalak J., 2005a: Terminologia polska w zakresie technologii interoperacyjnych w geomatyce. [W:] (red.) Gaździcki J. Internetowy Leksykon Geomatyczny.
URL: http://www.ptip.org.pl/auto.php?page=Encyclopedia&enc=1

Michalak J., 2005b: HGLML – HydroGeoLogical Markup Language – znacznikowy język wymiany geoinformacji hydrogeologicznej. Współczesne Problemy Hydrogeologii, t. XII: 499-504.

Michalak J., Nawalany M., Sadurski A., (red.) 2011: Schematyzacja warunków hydrogeologicznych na potrzeby numerycznego modelowania przepływu w JCWPd. Wyd. PIG – PIB, Warszawa.
URL: http://www.psh.gov.pl/plik/id,6091,v,artykul_4556.pdf

Michalak J, 2012: Testowanie roboczych wersji specyfikacji danych tematów załączników II i III INSPIRE. *Roczniki Geomatyki*, t. 10, z. 2, PTIP, Warszawa.

Nowicki B., Staniszkis W., 2002: Inteligentny system zarządzania wiedzą – prezentacja projektu. [W:] Mat. Konferencji eDemocracy, VI Konf. Miasta w Internecie, Zakopane.

OMG (Object Management Group), 2001: OMG Unified Modeling Language Specification, version 1.4. OMG Document Repository. URL: http://cgi.omg.org/docs/formal/01-09-67.pdf

OMG (Object Management Group), 2003: Object Management Group, Model Driven Architecture Guide Version 1.0.1 URL: http://www.omg.org/cgi-bin/doc?omg/03-06-01.pdf

OMG (Object Management Group), 2010: Object Constraint Language.
URL: http://www.omg.org/spec/OCL/2.2

Pachelski W., Parzyński Z. ,2007: Aspekty metodyczne wykorzystania norm serii ISO 19100 do budowy geodezyjnych składników krajowej infrastruktury danych przestrzennych. *Roczniki Geomatyki*, t.5, z.3, PTIP, Warszawa.

Peng Z_R., Zhang C., 2004: The roles of geography markup language (GML), scalable vector graphics (SVG), and Web feature service (WFS) specifications in the development of Internet geographic information systems (GIS).
URL: http://gis.geog.uconn.edu/personal/paper1/journal%20paper/3%202004%20GeographicalSystem1.pdf

PKN (Polski Komitet Normalizacyjny), 2010: Informacje podstawowe o PN.
URL: http://www.pkn.pl/informacje-podstawowe-o-pn

Płoski Z., 1999: Słownik Encyklopedyczny – Informatyka. Wyd. Europa, Warszawa.

Portele C., 2007: OpenGIS Geography Markup Language (GML) Encoding Standard. OpenGIS Standard.
URL: http://portal.opengeospatial.org/files/?artifact_id=20509

Portele C., 2008a: Mapping UML to GML Application Schemas. Guidelines and Encoding Rules. Interactive Instruments GmbH.
URL: http://www.interactive-instruments.de/ugas/UGAS-Guidelines-and-Encoding-Rules.pdf.

Portele C., 2008b: Mapping UML to GML Application Schemas. ShapeChange – Architecture and Description. Interactive Instruments GmbH. URL: http://www.interactive-instruments.de/ugas/ShapeChange.pdf

Portele C., 2012: OGC Geography Markup Language (GML) – Extended schemas and encoding rules. OpenGIS Implementation Standard. URL: https://portal.opengeospatial.org/files/?artifact_id=46568

Refsgaard J. C., Henriksen H. J., 2004: Modelling guidelines – terminology and guiding principles. *Advances in Water Resources* 27 (2004): 71-82.

Schmuller J., 2003: UML dla każdego. Helion, Gliwice.

Ustawa z dnia 12 września 2002 r. o normalizacji.
URL: http://www.pkn.pl/sites/default/files/ustawa_o_normalizacji_2.pdf

Ustawa z dnia 4 marca 2010 r. o infrastrukturze informacji przestrzennej, Dz.U. 2010 Nr 76, poz. 489.
Skogan D., 1999: UML as a Schema Language for XML based data Interchange. Materiały konferencji UML'99. URL: http://xml.coverpages.org/skoganUMLpaper-pdf.gz
Sparx System, Enterprise Architect. URL: http://www.sparxsystems.com.au
Subieta K., 1998: Obiektowość w projektowaniu i bazach danych. Akademicka Oficyna Wydawnicza PLJ, Warszawa.
Subieta K., 1999a: Słownik terminów z zakresu obiektowości. Akademicka Oficyna Wyd. PLJ, Warszawa. URL: http://www.ipipan.waw.pl/~subieta/artykuly/slownik_obiektowosci/hasla_slownika.html
Subieta K., 1999b: Wprowadzenie do obiektowych metodyk projektowania i notacji UML. Jedenasta Górska Szkoła PTI Szczyrk.
Tennakoon W., 2003: Visualization of GML data using XSLT. URL: http://www.itc.nl/library/Papers_2003/msc/gim/tennakoon.pdf
TWG GE (INSPIRE Thematic Working Group – Geology), 2011: D2.8.II.4 INSPIRE Data Specification on Geology – Draft Guidelines. Version 2.9.1. URL: http://inspire.jrc.ec.europa.eu/documents/Data_Specifications/INSPIRE_DataSpecification_GE_v2.01.pdf
TWG-CP (INSPIRE Thematic Working Group – Cadastral Parcels), 2010: D2.8.I.6 INSPIRE Data Specification on Cadastral Parcels – Guidelines, version: 3.0.1. URL: http://inspire.jrc.ec.europa.eu/documents/ Data_Specifications/INSPIRE_DataSpecification_CP_v3.0.1.pdf
WFD WG GIS (Working Group 3.1 – GIS), 2003: Guidance Document No 9 – Implementing the Geographical Information System Elements (GIS) of the Water Framework Directive. Water Framework Directive (WFD) – Common Implementation Strategy. URL: http://www.ec-gis.org/docs/F2305/GIS-GD.PDF
Woolf A., 2009: Enterprise Architect instructions, STFC Rutherford Appleton Laboratory. URL: http://wiki.services.eoportal.org/tiki-download_wiki_attachment.php?attId=732
Zhang C., Peng Z-R., Li W., Day M. J., 2003: GML-Based Interoperable Geographical Databases. URL: http://www.ucgis.org/summer03/studentpapers/chuanrongzhang.pdf