POLSKIE TOWARZYSTWO INFORMACJI PRZESTRZENNEJ
ROCZNIKI GEOMATYKI 2013 ❍ Tom XI ❍ Zeszyt 1(58)

NIEJEDNOZNACZNOŚĆ TRANSFORMACJI UML-GML

UML-GML TRANSFORMATION AMBIGUITY

Agnieszka Chojka*

Uniwersytet Warmińsko-Mazurski w Olsztynie, Wydział Geodezji i Gospodarki Przestrzennej

Słowa kluczowe: UML, GML, kodowanie, niejednoznaczność, anomalia
Keywords: UML, GML, encoding, ambiguity, anomaly

Wstęp

Uchwalenie w Polsce ustawy *o infrastrukturze informacji przestrzennej*, która jest transpozycją dyrektywy INSPIRE (przystosowaniem przepisów dyrektywy do prawa krajowego) spowodowało konieczność nowelizacji wielu ustaw i przepisów prawnych, w tym ustawy *Prawo geodezyjne i kartograficzne*. Postanowiono zastąpić instrukcje i wytyczne (często już przestarzałe) rozporządzeniami Rady Ministrów lub odpowiedniego ministra, które z jednej strony stanowią załączniki do ustawy *Prawo geodezyjne i kartograficzne*, a z drugiej strony wprowadzają w życie niektóre zalecenia dyrektywy INSPIRE.

Integralną częścią opracowywanych w Głównym Urzędzie Geodezji i Kartografii rozporządzeń są schematy aplikacyjne UML (definiują strukturę informacyjną baz danych, właściwych dla danego rozporządzenia) oraz schematy aplikacyjne GML. Jednak, mimo iż schematy te zostały przygotowane zgodnie z normami ISO serii 19100 w dziedzinie informacji geograficznej, co ma zapewnić interoperacyjność opracowanych na ich podstawie zbiorów danych przestrzennych, w trakcie ich tworzenia napotkano wiele problemów technicznych związanych z transformacją UML-GML.

W artykule, na przykładzie schematów aplikacyjnych UML i GML opracowanych w ramach prac GUGiK związanych z implementacją postanowień dyrektywy INSPIRE dla potrzeb służby geodezyjnej i kartograficznej, omówiono niektóre niejednoznaczności związane z przekształcaniem schematów aplikacyjnych UML na odpowiadające im schematy aplikacyjne GML oraz związane z tym błędy, anomalia i nieprawidłowości. Rozważono również, jak w przyszłości zaradzić tego typu niejednoznacznościom, jak zapewnić jednoznaczną drogę przekształcania UML na GML.

---

* Autorka od 2010 r. bierze udział w pracach związanych ze wsparciem działań GUGiK w zakresie tworzenia standardów technicznych dotyczących danych przestrzennych, w tym opracowanie schematów aplikacyjnych GML dla projektów rozporządzeń oraz weryfikacja pod względem merytorycznym i formalnym opracowanych schematów aplikacyjnych UML i GML.

Języki oraz schematy aplikacyjne UML i GML

Język UML (ang. *Unified Modeling Language*) to zunifikowany (ujednolicony) graficzny język modelowania. W dziedzinie informatyki wykorzystywany do opisu świata obiektów w analizie obiektowej i programowaniu obiektowym, ale również stosowany do wymiany informacji o systemach i oprogramowaniu za pomocą diagramów oraz uzupełniającego je tekstu. Za pomocą UML można, m.in. definiować wymagania, projektować architekturę rozwiązań, modelować struktury danych. Poza opisem struktur statycznych systemu (diagramy statyczne) istnieje również możliwość przedstawienia jego zachowania (diagramy dynamiczne). W dziedzinie geoinformatyki UML stanowi środek formalny modelowania informacji geograficznej i służy do opisu świata obiektów rzeczywistych. Zalecany jest także przez normy ISO serii 19100, jako język schematu pojęciowego (ang. *conceptual schema language*). Jednak z całego bogactwa języka UML w dziedzinie modelowania informacji geograficznej wykorzystuje się przede wszystkim możliwości modelowania obiektowego oferowane przez diagramy klas (w modelach uwzględnia się głównie klasy z atrybutami, bez metod) i pakietów.

Schemat aplikacyjny UML (ang. UML *application schema*) to model definiujący pojęcia z pewnej dziedziny (przestrzeni rozważań, przedmiotu zainteresowań), zapisany za pomocą oznaczeń klas i powiązań między nimi (diagram klas), właściwych dla języka UML. Stanowi on opis struktur logicznych danych przestrzennych oraz opis semantyki ich zawartości. Co więcej, jest to opis niezależny od platformy sprzętowo-programowej.

Schemat aplikacyjny UML powinien być zapisany w języku schematu pojęciowego UML według zasad określonych w standardach ISO/TS 19103 (ISO/TC 211, 19103:2005) i ISO 19109 (ISO/TC 211, 19109:2009) oraz składać się z pojęć określonych przez dziedzinę zastosowań, wyrażonych jako klasy i powiązania między nimi. Niektóre z klas mogą być zaimportowane ze schematów znormalizowanych z innych standardów, tzn. schemat aplikacyjny opracowany przez użytkownika, poza klasami opisującymi daną dziedzinę zastosowań, może dodatkowo zawierać klasy pochodzące ze schematów aplikacyjnych zdefiniowanych w normach ISO serii 19100 lub innych dokumentach standaryzacyjnych.

Natomiast język GML (ang. *Geography Markup Language*) to język znaczników przeznaczony do opisu danych geograficznych. W geoinformatyce jest on stosowany jako język formalny do opisu struktur danych (zalecany przez normy ISO serii 19100) oraz jako otwarty format wymiany danych przestrzennych (wektorowych i opisowych) pomiędzy różnymi systemami geoinformacyjnym (GIS, ang. *Geographical Information System*), np. poprzez wykorzystanie jednej z usług geoinformacyjnych – usługi WFS (ang. *Web Feature Service*), która na żądanie dostarcza dane zakodowane w tym formacie.

GML jest aplikacją, czyli zastosowaniem języka XML (ang. *eXtensible Markup Language*). Określa regułę kodowania dla schematów aplikacyjnych zgodnych z normami ISO serii 19100, opartą na XML zgodnie z ISO 19118 (ISO/TC 211, 19118:2011), definiuje sposób zapisu w języku XML Schema (Schemat XML lub Schemat Rozszerzalnego Języka Znaczników) określonych właściwości przestrzennych i nieprzestrzennych (zdefiniowanych w normach ISO serii 19100) obiektów geograficznych, np. jednostki miary, geometria i topologia, systemy odniesienia.

Schemat aplikacyjny GML to, podobnie jak schemat aplikacyjny UML, model definiujący pojęcia z pewnego zakresu przedmiotowego, ale tym razem zapisany w języku XML Schema zgodnie z regułami określonymi w normie ISO 19136 (ISO/TC 211, 19136:2007). Dodatko-

wo schemat aplikacyjny GML powinien importować schemat GML, który składa się z komponentów w przestrzeni nazw XML http://www.opengis.net/gml/3.2, przeznaczonych do zapisu określonych właściwości przestrzennych i nieprzestrzennych obiektów geograficznych.

Jednym ze sposobów budowy schematu aplikacyjnego GML jest uprzednio opracowanie schematu aplikacyjnego UML z zastosowaniem reguł określonych w normie ISO 19109, a następnie przekształcenie go na odpowiadający mu schemat aplikacyjny GML, zapisany w języku XML Schema, zgodnie z regułami kodowania określonymi w normach ISO 19118 i ISO 19136. Taki sposób tworzenia schematów aplikacyjnych GML przyjęto również w GUGiK, w ramach przygotowania rozporządzeń.

Przeznaczenie UML i GML

W zakresie budowy infrastruktur informacji przestrzennej języki UML i GML wykorzystywane są do opracowania schematów aplikacyjnych, odpowiednio UML i GML. Jednak każdy z tych języków ma nieco inne przeznaczenie. Ponieważ język UML jest językiem graficznym, stanowi zbiór oznaczeń (w tym piktogramów) i diagramów, wykorzystywanych do specyfikowania, konstruowania, wizualizacji i dokumentowania elementów systemów informatycznych, dlatego w głównej mierze przeznaczony jest dla ludzi. Stanowi środek i platformę komunikacji dla przyszłych użytkowników systemu, menadżerów, analityków, architektów, projektantów, programistów i testerów.

Natomiast język GML to język znaczników (ang. *markup*), którego wyrażenia zapisywane są za pomocą specyficznego kodu. Są one dedykowane głównie do przetwarzania maszynowego, a więc przez aplikacje komputerowe umiejące interpretować kod GML i prezentować go w formie przyjaznej dla człowieka. GML nie jest językiem przeznaczonym do bezpośredniej konfrontacji z użytkownikiem.

Język GML bazuje na gramatyce języka XML, który jest językiem uniwersalnym, a przede wszystkim rozszerzalnym. Użytkownik może definiować swoje własne znaczniki, a konkretne struktury danych mogą być zapisywane na wiele równoważnych sposobów (patrz: ramka poniżej). Jest to zaletą języka XML – dzięki temu jest to język niezależny od platformy sprzętowo-programowej, ale i jego wadą – taka uniwersalność prowadzi do niejednoznaczności w interpretacji poszczególnych struktur, a tym samym do różnych błędów i anomalii.

Poniżej zamieszczono przykład zapisu informacji o płci osoby za pomocą atrybutu (kolumna lewa) i/lub za pomocą elementu (kolumna prawa) w języku XML.

```
<osoba płeć="kobieta">
  <imię>Anna</imię>
  <nazwisko>Mazurek</nazwisko>
</osoba>
```

```
<osoba>
  <płeć>kobieta</płeć>
  <imię>Anna</imię>
  <nazwisko>Mazurek</nazwisko>
</osoba>
```

Oba przedstawione sposoby kodowania danych są równoważne, tzn. przenoszą ten sam zbiór informacji dotyczących osoby. Jednak przyjmuje się, że dla danych najlepiej używać elementów, natomiast dla informacji, które stanowią dodatkową charakterystykę danych (np. nazwa układu współrzędnych dla podanych wewnątrz współrzędnych), najlepiej stosować atrybuty (W3schools, 2013).

Transformacja UML-GML

Transformacja schematu aplikacyjnego UML, na odpowiadający mu schemat aplikacyjny GML, może być zrealizowana metodą automatyczną lub metodą ręczną (Michalak, Chojka, Zwirowicz-Rutkowska, Parzyński, 2012).

Metoda automatyczna wymaga przede wszystkim odpowiedniego oprogramowania, które umożliwia automatyczne wygenerowanie schematu aplikacyjnego GML ze schematu aplikacyjnego UML. Obecnie na rynku dostępne są dwa takie narzędzia, oba bezpłatne: *ShapeChange* (Portele, 2008a; Portele, 2008b) i *FullMoon* (Githaiga, 2010; Cox, 2011). Jednak wymagają one wykorzystania dodatkowo komercyjnego oprogramowania *Enterprise Architect* firmy Sparx Systems (Sparx Systems, 2013), pozwalającego na właściwe przygotowanie wyjściowego schematu aplikacyjnego UML (wykorzystanie pliku profilu UML z odpowiednimi metkami, ang. *tagged values*). Ponadto dla różnych elementów schematu aplikacyjnego UML można dodatkowo określić metki, które umożliwiają kontrolę przekształcenia schematu aplikacyjnego UML na GML, czyli generowanie plików XSD zapisanych w języku XML Schema.

Zarówno *ShapeChange*, jak i *FullMoon*, wymagają zastosowania specjalnego szablonu UML do opracowania schematu aplikacyjnego UML, który ma zostać automatycznie przekształcony na schemat aplikacyjny GML. Szablon ten (profil UML) został przygotowany w postaci pliku XML i zawiera standardowe stereotypy oraz metki określone przez normę ISO 19136. Można go wykorzystać jedynie w oprogramowaniu *Enterprise Architect*, które stanowi zaawansowane narzędzie do modelowania systemów informatycznych za pomocą języka UML.

Metoda ręczna wymaga przede wszystkim bardzo dobrej znajomości norm ISO serii 19100, w szczególności ISO/TS 19103 i ISO 19109, które określają zasady budowy schematów aplikacyjnych UML oraz normy ISO 19136 załącznik E, w którym zdefiniowano regułę kodowania schematów aplikacyjnych UML na GML. W metodzie tej wystarczy opracować schemat aplikacyjny UML w dowolnym narzędziu wspomagającym tworzenie diagramów klas UML, a następnie korzystając z normy ISO 19136 napisać kod GML odpowiadający schematowi aplikacyjnemu UML, najlepiej korzystając z oprogramowania (bezpłatnego lub komercyjnego), które wspomaga tworzenie plików XSD (XML Schema) i dokonuje ich walidacji, czyli kontroli poprawności składniowej. Przykładem takiego oprogramowania (komercyjnego) są *XMLSpy* firmy Altova (Altova, 2013) oraz *oXygen XML Editor* firmy SyncRO Soft Ltd. (SyncRO Soft Ltd., 2013), które stanowią jedne z najbardziej wszechstronnych na rynku narzędzi do projektowania zaawansowanych aplikacji XML.

Przekształcenie schematu aplikacyjnego UML, zgodnego z ISO 19109, na odpowiadający mu schemat aplikacyjny GML, oparte jest na zbiorze reguł kodowania określonych w ISO 19136. Zasady te podano w załączniku E tej normy i oparto na ogólnym założeniu, że definicja klasy w schemacie aplikacyjnym UML jest przekształcana na deklarację typu i elementu w schemacie aplikacyjnym GML (zapisanym w języku XML Schema) według zależności podanych w tabeli.

Do utworzenia schematów aplikacyjnych UML, stanowiących element składowy rozporządzeń opracowanych przez GUGiK – m.in. rozporządzenia: *w sprawie ewidencji miejscowości, ulic i adresów* (EMUiA, 2012), *w sprawie bazy danych geodezyjnej ewidencji sieci uzbrojenia terenu, bazy danych obiektów topograficznych oraz mapy zasadniczej* (GESUT,

Tabela. Reguły transformacji UML-GML (źródło: ISO/TC 211, ISO 19136:2007)

Schemat aplikacyjny UML	Schemat aplikacyjny GML
Pakiet	Jeden dokument XML Schema na pakiet
<<Application Schema>>	Dokument XML Schema
<<DataType>>	Element globalny, którego modelem zawartości jest element *complexType* w XML Schema o zakresie globalnym, typ właściwości
<<Enumeration>>	Ograniczenie *xsd:string* z wartościami wyliczenia
<<CodeList>>	Odwołanie do słownika lub alternatywnie unia wyliczenia i wzorca
<<Union>>	Grupa wyboru, której członkami są obiekty odpowiadające *FeatureTypes* lub *DataTypes*
<<FeatureType>>	Element globalny, którego modelem zawartości jest typ XML Schema o zakresie globalnym, pochodzący z bezpośredniego lub pośredniego rozszerzenia gml: *AbstractFeatureTyp*e, typ właściwości
Brak stereotypu lub <<Type>>	Element globalny, którego modelem zawartości jest typ XML Schema o zakresie globalnym, pochodzący z bezpośredniego lub pośredniego rozszerzenia gml: *AbstractGMLTyp*e, typ właściwości
Operacje	Niekodowane
Atrybut	Lokalny *xsd:element,* typ jest również typem właściwości (jeśli typ jest typem złożonym) lub typem prostym
Rola powiązania (asocjacji)	Lokalny *xsd:elemen*t, typ jest zawsze typem właściwości (tylko role nazwane i nawigowalne)
Ograniczenia OCL	Niekodowane

BDOT, MZ, 2013), *w sprawie państwowego rejestru granic i powierzchni jednostek podziałów terytorialnych kraju* (PRG, 2012) – bardzo słusznie użyto aplikacji *Enterprise Architect*, ale niestety nie wykorzystano szablonu UML, czyli pliku profilu UML z odpowiednimi metkami, uniemożliwiając tym samym automatyczne wygenerowanie schematu aplikacyjnego GML. Stąd też schematy aplikacyjne GML zostały opracowane metodą ręczną (przy użyciu oprogramowania *XMLSpy*), gdzie niestety dużą rolę odgrywał czynnik ludzki, a więc nie trudno było o błędy składniowe i logiczne.

Należy pamiętać, iż schemat aplikacyjny GML definiuje możliwą strukturę danych, a więc sposób uporządkowania konkretnych danych w pliku GML (XML). Dlatego też błędne zapisy struktur danych mają bezpośredni wpływ na możliwości generowania plików GML z konkretnymi danymi (obiektami), mogą być przyczyną różnych problemów i anomalii na etapie produkcji danych.

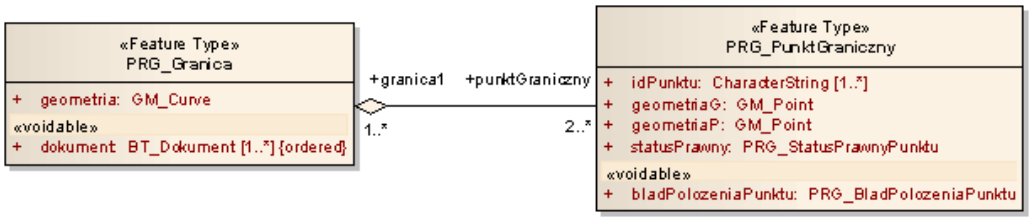
Problemy

Najczęstszym problemem, jaki pojawia się podczas transformacji schematów aplikacyjnych UML na odpowiadające im schematy aplikacyjne GML, jest właściwe odwzorowanie związków między klasami. Jeżeli powiązanie między dwiema klasami nie zostanie opisane przynajmniej jedną rolą w UML, wówczas nie może zostać odwzorowane w języku GML. Zatem informacja o powiązaniu między klasami w UML zostanie pominięta w XML Schema.

Ponadto poszczególne rodzaje powiązań (np. nawigacje, agregacje) nie są rozróżniane w GML. Jeśli z kolei powiązanie zostanie opisane dwiema rolami w UML, wówczas dodatkowo w kodzie GML istnieje możliwość wprowadzenia informacji o nazwie roli po przeciwnej stronie powiązania (ramka poniżej, element *gml:reversePropertyName*, fragment schematu aplikacyjnego GML *PRG.xsd* (PRG, 2012)).

W kodzie GML rola charakteryzująca powiązanie UML (rys. 1) jest zapisywana w typie złożonym jako element lokalny (w poniższym przykładzie: *punktGraniczny*), czyli stanowi odpowiednik atrybutu klasy w UML.

```
<complexType name="PRG_GranicaType">
 <complexContent>
  <extension base="prg:PRG_ObiektOgolnyType">
   <sequence>
    <element name="geometria" type="gml:CurvePropertyType"/>
    <element name="dokument" maxOccurs="unbounded">
     <complexType>
      <complexContent>
       <extension base="bt:BT_DokumentPropertyType">
        <attribute ref="gco:nilReason"/>
       </extension>
      </complexContent>
     </complexType>
    </element>
    <!-- Roles -->
    <element name="punktGraniczny" type="prg:PRG_PunktGranicznyPropertyType" minOccurs="2"
maxOccurs="unbounded">
     <annotation>
      <appinfo>
       <gml:reversePropertyName>prg:granica1</gml:reversePropertyName>
      </appinfo>
     </annotation>
    </element>
   </sequence>
  </extension>
 </complexContent>
</complexType>
```



Rys. 1. Powiązanie między klasami w UML opisane rolami (źródło: PRG, 2012)

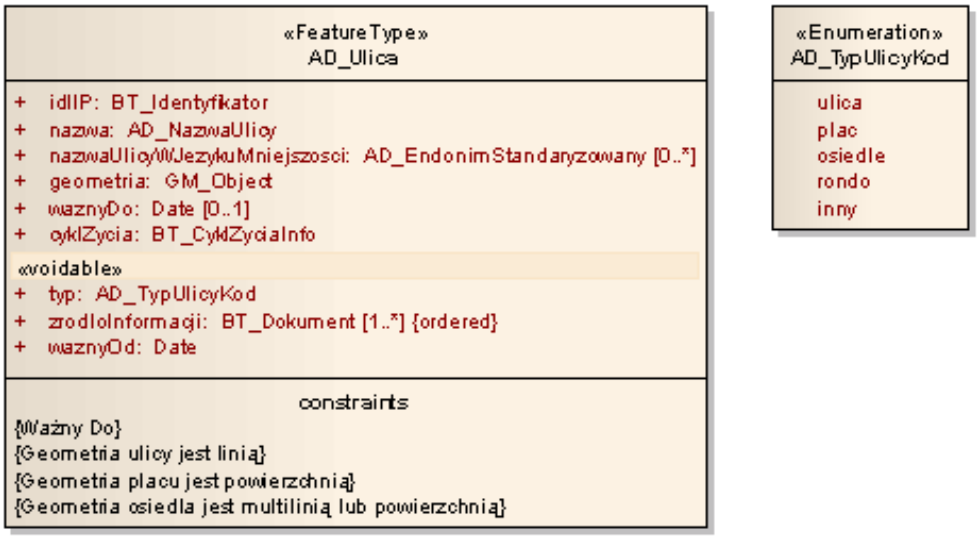
Dodatkowo język UML pozwala na wprowadzenie stereotypu «voidable», który jest przypisany do atrybutu klasy lub roli w powiązaniu na schemacie aplikacyjnym UML (np. atrybut *dokument* na rys. 1). Stereotyp ten oznacza, że jest to atrybut specjalny i ma on zastosowanie, gdy pewna właściwość (cecha) obiektu przestrzennego nie jest prezentowana w zbiorze danych przestrzennych, ale może być obecna lub mieć zastosowanie w świecie rzeczywistym. Odpowiednikiem stereotypu «voidable» w schemacie aplikacyjnym GML jest atrybut

nilReason (patrz: ramka powyżej), który w pliku GML z konkretnymi danymi pozwala na zapisanie informacji o przyczynie braku wartości dla atrybutu specjalnego, np. *inapplicable* (nie stosuje się), *missing* (brak danych), *template* (tymczasowy brak danych), *unknown* (wartość nieznana), *withheld* (wartość zastrzeżona).

Okazuje się jednak, że właściwe zakodowanie w GML atrybutu specjalnego jest bardzo istotne. W przeciwnym wypadku może to prowadzić do anomalii przy generowaniu plików GML z konkretnymi obiektami.

Niejednoznaczności i anomalie

Przykładem niejednoznaczności zapisu struktur danych w GML (XML Schema) może być zakodowanie atrybutu oznaczonego w UML stereotypem «voidable», np. atrybut *typ* w klasie *AD_Ulica* (schemat aplikacyjny *AD_EMUiA.xsd* (EMUiA, 2012)). Atrybut ten jest typu prostego wyliczeniowego *AD_TypUlicyKod*, w schemacie aplikacyjnym UML oznaczony stereotypem «Enumeration» (rys. 2).



Rys. 2. Klasa posiadająca atrybut typu wyliczeniowego, oznaczony stereotypem «voidable» (źródło: EMUiA, 2012)

Taki atrybut w XML Schema można zapisać na dwa sposoby:

1.

```
<element name="typ" nillable="true">
  <complexType>
    <simpleContent>
      <extension base="mua:AD_TypUlicyKodType">
        <attribute name="nilReason" type="gml:NilReasonType"/>
      </extension>
    </simpleContent>
  </complexType>
</element>
```

2. (taki wariant kodowania zastosowano w schemacie aplikacyjnym GML *AD_EMUiA.xsd*)

```
<element name="typ">
 <complexType>
  <simpleContent>
   <extension base="mua:AD_TypUlicyKodType">
    <attribute ref="gco:nilReason"/>
   </extension>
  </simpleContent>
 </complexType>
</element>
```

Niestety, jeżeli atrybut jest typu prostego wyliczeniowego (w UML klasa ze stereotypem «Enumeration») i dodatkowo jest atrybutem wymaganym (minimalna liczba jego wystąpień wynosi 1), przypadek drugi prowadzi do poniższej anomalii:

```
<mua:AD_Ulica gml:id="ID_Ulica_01">
 …
 <mua:typ gco:nilReason="missing">plac</mua:typ>
 …
</mua:AD_Ulica>
```

W pliku GML z konkretnymi danymi, oprócz podania informacji o przyczynie braku wartości dla atrybutu (tutaj: *missing*), należy mimo wszystko przypisać elementowi konkretną wartość wyliczenia. W przeciwnym wypadku plik GML nie przechodzi walidacji. Jest to anomalia, ponieważ zastosowanie atrybutu *nilReason* powinno właśnie umożliwić niepodanie wartości atrybutu.

Poprawny przykład pliku GML (zgodny z pierwszym wariantem zapisu atrybutu specjalnego ze stereotypem «voidable») powinien wyglądać następująco:

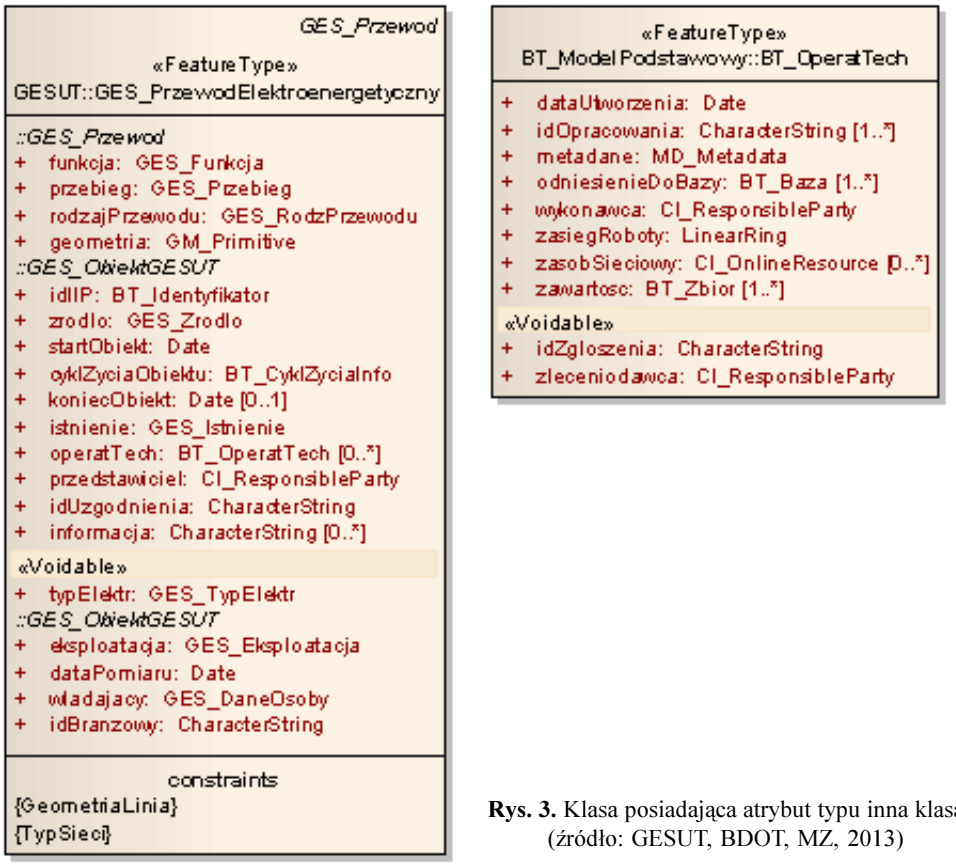
```
<mua:AD_Ulica gml:id="ID_Ulica_01">
 …
 <mua:typ xsi:nil="true" nilReason="missing"/>
 …
</mua:AD_Ulica>
```

Walidacja i błędy

Walidacja plików GML, w tym dokumentów XSD i XML, polega na sprawdzeniu zgodności składni tych dokumentów ze standardami wyznaczonymi przez Konsorcjum W3C (ang. *World Wide Web Consortium*). Należy pamiętać, że walidator wychwytuje jedynie błędy gramatyczne (składniowe), błędy logiczne nie podlegają kontroli. Zatem wykorzystując metodę ręczną transformacji UML-GML nie trudno o błąd logiczny, który może powodować wiele anomalii na etapie generowania plików GML z konkretnymi obiektami.

Zgodnie z normą ISO 19136, jeżeli atrybut klasy UML jest typu złożonego (inną klasą), np. w klasie *GES_PrzewodElektroenergetyczny*, atrybut *operatTech* jest typu *BT_OperatTech* (rys. 3),

wtedy w schemacie aplikacyjnym GML, do elementu *operatTech* należy przypisać typ z przyrostkiem *„PropertyType" (BT_OperatTechPropertyType)*. Jeśli przez pomyłkę zostanie przypisany typ z przyrostkiem *„Type" (BT_OperatTechType)*, wówczas w pliku GML zostanie zagubiona informacja o tym, że *BT_OperatTech* jest niezależnym bytem (klasą w UML, typem złożonym w XML), a tym samym można się do niego odnieść przez referencję, a niekoniecznie kopiować całą jego strukturę (ma to zastosowanie tylko w przypadku klas ze stereotypem «FeatureType»). Użycie typu *„PropertyType"* pozwala uniknąć redundancji danych (niepożądana cecha bazy danych), a co za tym idzie znacznie zmniejszyć rozmiary plików GML, ponieważ dane dotyczące przewodów elektroenergetycznych mogą być zapisane w jednym pliku, a dane charakteryzujące operaty techniczne w innym.



Rys. 3. Klasa posiadająca atrybut typu inna klasa (źródło: GESUT, BDOT, MZ, 2013)

Jeżeli w schemacie aplikacyjnym GML (w pliku XSD) do atrybutu *operatTech* przypisano typ złożony *BT_OperatTechPropertyType* (taki sposób kodowania zastosowano w schemacie aplikacyjnym *GES_GESUT.xsd* (GESUT, BDOT, MZ, 2013)), wtedy przykładowe dane w pliku GML mogłyby zostać zapisane następująco:

```
<ges:GES_PrzewodElektroenergetyczny gml:id="ID_PrzewodElektroenergetycznych_5678">
  ...
  <ges:operatTech>
    <bt:BT_OperatTech gml:id="ID_OperatTech_2002.10">
      <bt:dataUtworzenia>2002-10-15</bt:dataUtworzenia>
      <bt:idOpracowania>2002.10</bt:idOpracowania>
      <bt:metadane
xlink:href="urn:gugik:specyfikacje:gmlas:geodezyjnaEwidencjaSieciUzbrojeniaTerenu#Metadane_GESUT"/>
      <bt:odniesienieDoBazy>GESUT</bt:odniesienieDoBazy>
      <bt:wykonawca
xlink:href="urn:gugik:specyfikacje:gmlas:geodezyjnaEwidencjaSieciUzbrojeniaTerenu#ID_Wykonawca_09876"/>
      <bt:zasiegRoboty>
      ...
      </bt:zasiegRoboty>
      <bt:zawartosc>
      ...
      </bt:zawartosc>
      <bt:idZgloszenia>2002.08.OP</bt:idZgloszenia>
      <bt:zleceniodawca
xlink:href="urn:gugik:specyfikacje:gmlas:geodezyjnaEwidencjaSieciUzbrojeniaTerenu#ID_Zleceniodawca_119899"/>
    </bt:BT_OperatTech>
  </ges:operatTech>
  ...
</ges:GES_PrzewodElektroenergetyczny>
```

Lub zamiast wczytywania całej struktury *BT_OperatTech*, tylko referencja do konkretnego obiektu poprzez jego identyfikator *ID_OperatTech_2002.10*:

```
<ges:GES_PrzewodElektroenergetyczny gml:id="ID_PrzewodElektroenergetycznych_5678">
  ...
  <ges:operatTech
xlink:href="urn:gugik:specyfikacje:gmlas:geodezyjnaEwidencjaSieciUzbrojeniaTerenu#ID_OperatTech_2002.10"/>
  ...
</ges:GES_PrzewodElektroenergetyczny>
```

Niestety powyższy sposób odwołania się jednego obiektu do drugiego jest niemożliwy do zastosowania w przypadku rozporządzenia dotyczącego GESUT, ponieważ w schemacie aplikacyjnym *GML BT_ModelPodstawowy.xsd*, będącym częścią tego opracowania, błędnie zdefiniowano typ *BT_OperatTechPropertyType*. Element <sequence> powinien posiadać wskaźnik występowania minOccurs, pozwalający na określenie minimalnej ilości wystąpień tego elementu (tylko dla klas ze stereotypem «FeatureType»), w tym przypadku: <sequence minOccurs=”0”>. Brak takiego wskaźnika prowadzi do kolejnej anomalii – brak możliwości wykorzystania referencji, a tym samym nadmiarowość danych.

Natomiast, jeśli w schemacie aplikacyjnym GML do atrybutu *operatTech* przypisano typ złożony *BT_OperatTechType*, wtedy przykładowe dane w pliku GML mogłyby zostać zapisane następująco:

```
<ges:GES_PrzewodElektroenergetyczny gml:id="ID_PrzewodElektroenergetycznych_5678">
  ...
  <ges:operatTech gml:id="ID_OperatTech_2002.10">
    <bt:dataUtworzenia>2002-10-15</bt:dataUtworzenia>
    <bt:idOpracowania>2002.10</bt:idOpracowania>
    <bt:metadane
xlink:href="urn:gugik:specyfikacje:gmlas:geodezyjnaEwidencjaSieciUzbrojeniaTerenu#Metadane_GESUT"/>
    <bt:odniesienieDoBazy>GESUT</bt:odniesienieDoBazy>
    <bt:wykonawca
xlink:href="urn:gugik:specyfikacje:gmlas:geodezyjnaEwidencjaSieciUzbrojeniaTerenu#ID_Wykonawca_09876"/>
    <bt:zasiegRoboty>
     ...
    </bt:zasiegRoboty>
    <bt:zawartosc>
     ...
    </bt:zawartosc>
    <bt:idZgloszenia>2002.08.OP</bt:idZgloszenia>
    <bt:zleceniodawca
xlink:href="urn:gugik:specyfikacje:gmlas:geodezyjnaEwidencjaSieciUzbrojeniaTerenu#ID_Zleceniodawca_119899"/>
  </ges:operatTech>
  ...
</ges:GES_PrzewodElektroenergetyczny>
```

Wówczas wczytywana jest cała struktura elementów charakteryzujących typ złożony *BT_OperatTechType* i również nie ma możliwości alternatywnego zapisu poprzez referencję.

Wnioski

Należy pamiętać, że język GML jest formatem wymiany danych i nie jest to język przeznaczony do bezpośredniej konfrontacji z użytkownikiem. GML ma zapewnić interoperacyjną wymianę danych przestrzennych między różnymi systemami geoinformacyjnymi, jest to więc środek komunikacji między narzędziami.

Użytkownikowi dedykowany jest „przyjemny dla oka", „obrazkowy" język UML, który z kolei nie nadaje się dla maszyn, tzn. nie jest zrozumiały dla systemów geoinformacyjnych, aby stać się dla nich formatem wymiany danych. Stąd konieczność transformacji „obrazkowego" języka UML na „znacznikowy" język GML. Warto jednak wspomnieć, iż istnieje standard XMI (ang. *XML Metadata Interchange*), który daje możliwość zapisu schematów aplikacyjnych UML w języku XML, ale pełni on tylko rolę pośrednika podczas wymiany modeli UML między różnymi narzędziami.

Sama transformacja schematów aplikacyjnych UML na schematy aplikacyjne GML, opracowywane w GUGiK, wymaga jeszcze dopracowania i doprecyzowania, aby w przyszłości uniknąć niejednoznaczności zapisów struktur danych i nie powodować problemów podczas generowania plików GML z baz danych oraz nie wprowadzać w błąd użytkowników. Dlatego rekomenduje się ustalenie zbioru jednoznacznych metod transformacji UML-GML, m.in. określenie sposobu kodowania atrybutów specjalnych (oznaczonych w UML stereotypem «voidable») oraz stosowanie tylko kodowania poprzez referencję w przypadku ról powiązań i atrybutów typu inna klasa (stosowanie typu *gml:ReferenceType* zamiast typu z przyrostkiem *„PropertyType"*), aby zapobiec redundancji danych w plikach GML z konkretnymi obiektami. Należy także podkreślić, iż schematy aplikacyjne UML są tylko etapem pośrednim, właściwym celem są schematy aplikacyjne GML (pliki XSD), które wykorzystywane są w praktyce, na etapie produkcji plików GML z konkretnymi danymi.

Ponadto, aby w przyszłości zaradzić różnego rodzaju niejednoznacznościom i anomaliom, jak również zapewnić jednoznaczną drogę przekształcenia UML na GML, do opracowania schematów aplikacyjnych GML zaleca się zastosowanie łącznie języka XML Schema i języka Schematron, w którym zapisuje się asercje dotyczące dokumentu XML, czyli warunki, które muszą być spełnione w pewnym kontekście. Wówczas w schemacie XML Schema można zamodelować strukturę danych, zaś w schemacie Schematron – reguły kontekstowe.

Literatura

Altova, 2013: XMLSpy, http://www.altova.com/xml-editor/

Michalak J., Chojka A., Zwirowicz-Rutkowska A., Parzyński Z., 2012: Modele danych przestrzennych w UML i ich transformacja do schematów GML i struktur baz danych. Monografia. *Roczniki Geomatyki* t. 10, z. 1(51), PTIP Warszawa.

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

EMUiA, 2012: Rozporządzenie Ministra Administracji i Cyfryzacji z dnia 9 stycznia 2012 r. *w sprawie ewidencji miejscowości, ulic i adresów*. Dz.U. 2012 nr 0 poz. 125.

GESUT, BDOT, MZ, 2013: Rozporządzenie Ministra Administracji i Cyfryzacji z dnia 12 lutego 2013 r. *w sprawie bazy danych geodezyjnej ewidencji sieci uzbrojenia terenu, bazy danych obiektów topograficznych oraz mapy zasadniczej*. Dz.U. 2013 nr 0 poz. 383.

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

ISO/TC 211 (Geographic Information/Geomatics), Technical Specification 19103:2005, Geographic information – Conceptual schema language (Język schematów pojęciowych).

ISO/TC 211 (Geographic Information/Geomatics), ISO 19109:2009, Geographic information – Rules for application schema. Norma PN-EN ISO 19109:2009, Informacja geograficzna – Reguły schematów aplikacyjnych.

ISO/TC 211 (Geographic Information/Geomatics), ISO 19118:2011, Geographic information – Encoding. Norma PN-EN ISO 19118:2011, Informacja geograficzna – Kodowanie.

ISO/TC 211 (Geographic Information/Geomatics), ISO 19136:2007, Geographic information – Geography Markup Language (GML). Norma PN-EN ISO 19136:2009, Informacja geograficzna – Język znaczników geograficznych GML.

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

PRG, 2012: Rozporządzenie Rady Ministrów z dnia 10 stycznia 2012 r. *w sprawie państwowego rejestru granic i powierzchni jednostek podziałów terytorialnych kraju*. Dz.U. 2012 nr 0 poz. 199.

SparxSystem, 2013: Enterprise Architect, http://www.sparxsystems.com.au/

SyncRO Soft Ltd., 2013: oXygen XML Editor, http://www.oxygenxml.com/

W3Schools, 2013: W3Schools Online Web Tutorials, http://www.w3schools.com/

Abstract

Passing the "Spatial Information Infrastructure Law" in Poland, that is a transposition of the INSPIRE Directive, involved the necessity of many secondary acts and corresponding changes in other Laws, among others the „Geodetic and Cartographic Law". Decision was made to replace the existing instructions and guidelines by regulations of the Council of Ministers or relevant minister that, on the one hand, become annexes to the "Geodetic and Cartographic Law" and, on the other hand, implement the recommendations of the INSPIRE Directive.

An integral part of these regulations elaborated in the Head Office of Geodesy and Cartography in Poland are the UML and GML application schemas that define information structures of databases, corresponding to each regulation. Although these schemas were worked out according to the ISO 19100 series of International Standards in the Geographic Information (Geoinformation/Geomatics) domain, many technical problems connected with UML-GML transformation were identified during their preparation.

In this paper, on examples of UML and GML application schemas prepared in the Head Office of Geodesy and Cartography in Poland within the INSPIRE Directive implementation works, some ambiguities concerning UML to GML transformation were discussed as well as some errors and anomalies connected with this issues. Questions how to resolve this ambiguity and how to ensure single way in changing UML application schema into corresponding GML application schema were also considered.

dr inż. Agnieszka Chojka
agnieszka.chojka@uwm.edu.pl