*p*GML – estudo de um subconjunto ”Preciso”do GML2.12

Mário Ricardo de Novais Henriques

INESC-Porto, Rua Dr. Roberto Frias, n.378 4200-465 PORTO, Portugal,
ricardo.henriques@inescporto.pt,
http://www.inescporto.pt

Resumo A área dos Sistemas de Informação Geográfica (GIS) é das áreas da produção de *software* onde a necessidade de obter soluções sem falhas, dentro dos prazos e orçamentos é mais imperativa – não só pelo número de pessoas que directamente podem ser afectadas, como pelos custos e meios envolvidos. A par deste objectivo, há percepção de que estes sistemas dificilmente operam em máquinas modestas, dadas as descrições dos elementos do mundo real serem representadas por enormes colecções de entidades gráficas de baixo nível, e tal motiva a questão de saber se as ferramentas comerciais de GIS operam no nível correcto de abstracção.
É entre estes dois objectivos que surgiu o estudo de um subconjunto do GML, como forma de obter, por engenharia reversa, um modelo formal descrito numa notação conceituada e *standard*, o VDM-SL. Obtido o modelo, conseguiu-se uma representação abstracta e ao mesmo tempo capaz de ser utilizada como protótipo,[1] com o núcleo da funcionalidade para representação de um Sistema de Informação Geográfica, sobre o qual se pode raciocinar e derivar a implementação. A capacidade de raciocinar nesta fase é essencial de forma a evitar a propagação de erros e inconsistências do modelo que, a ser detectado só durante implementação final, provocariam atrasos e custos muito acrescidos.

Palavras-chave: Métodos formais, GML, GIS, VDM-SL.

1 Introdução

Há um quarto de século o eminente cientista da computação, John Backus escreveu [1] um artigo revolucionário que se tornaria um *ex-libris* da moderna ciência de computação. Na essência, o artigo proclamava obsoletas as linguagens de programação orientadas ao comando devida à sua ineficiência, por não serem independentes a alto nível, do padrão do modelo computacional subjacente – a arquitectura de Von Neumann. Alternativamente, o já na época existente estilo de programação funcional devia ser a opção, por duas razões: em primeiro lugar por potenciar a computação paralela e concorrente, e em segundo por ser fortemente baseada em estruturas algébricas. Backus salientou que apenas teriam sucesso as linguagens que possuíssem uma álgebra para raciocínio sobre os objectos que semanticamente descrevem. O impacto do primeiro argumento de Backus nas comunidades da ciência da computação e da computação paralela foi significativo.

[1] Através da utilização das VDM-Tools da IFAD, por exemplo.

Por contraste, o seu segundo argumento para alterar a computação foi em larga medida ignorado. De facto, apenas a minoria dos investigadores em *program calculi* seguiu esta direcção.

Num trabalho publicado em 1998 ficou demonstrado [12], na área do *software* de CAD, ser possível introduzir estruturas algébricas de alto nível capazes de abstrair entidades gráficas de baixo-nível como pontos, linhas e segmentos. Um CAD é um bom exemplo de uma tecnologia suportada pela computação, onde não se tinha evoluído muito na direcção do artigo de Backus, apesar de toda a evolução a nível de gráficos 3D, realidade virtual e multimédia.

Para além da representação da realidade, através de colecções de entidades gráficas de baixo nível, os GIS dispõem da capacidade de registo e análise topológicas. É natural transpor para o domínio dos GIS as mesmas preocupações:

- As ferramentas comerciais de GIS estão a trabalhar no nível correcto de abstracção?
- As descrições de elementos reais estão condenadas a serem representadas por imensas colecções de entidades gráficas (geométricas) de baixo nível?
- Ou possuem uma estrutura algébrica intrínseca passível de raciocínio?

No sentido de conseguir uma resposta a estas preocupações/questões, foi investigada a possibilidade de obter um modelo com o nível de abstracção considerado adequado, baseado no rigor dos métodos formais.

2 Contexto e objectivos

A propriedade mais importante de um sistema de modelação é a sua vocação para permitir a análise e raciocínio, aliando o rigor à abstracção. Nesse intuito foi utilizado o VDM-SL (*specification language*)[5], o qual resulta da evolução do VDM concebido nos laboratórios da IBM em Viena em 1973, cuja origem remonta a 1968, onde os participantes da *NATO Conference on Software Engineering* anteviram uma disciplina de desenvolvimento de *software* suportada por uma forte base científica.

Reconhecendo o GML (*Geography Markup Language*) como uma plataforma rica, aberta e que serve de denominador comum aos formatos proprietários para representação de informação geográfica, foi natural procurar por engenharia reversa obter um modelo na notação formal do VDM-SL e com isso investigar a possibilidade de obter um modelo capaz de responder às preocupações enunciadas na secção anterior. O resultado foi a obtenção do *p*GML (*precise* GML), mais conciso e enriquecido com invariantes topológicos que o GML por si só não retém.

A seguinte figura apresenta em (a) e (b) os elementos contextuais relevantes. (a) representa num sentido a **formalização** do GML, na qual se obtém o enriquecimento semântico do GML através da introdução de invariantes. No outro sentido há a exportação do modelo formal para o domínio do XML. Em (b) consegue-se a acomodação de "*domain specific languages*"de MF (Métodos Formais) aos GIS (*Geography Information System*). Tal traduz-se, num sentido, na formalização de um subconjunto do

domínio dos GIS numa notação formal e obtenção de um modelo – neste caso o *p*GML (*precise*GML). No sentido inverso abre-se o caminho para a obtenção de novos paradigmas GIS mediante a obtenção de novas derivações do modelo formal.

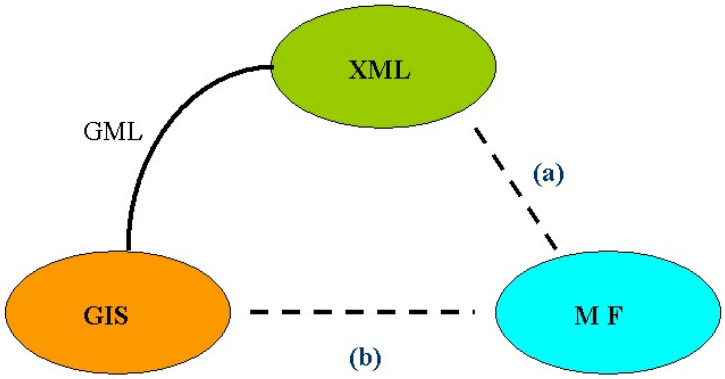

Figura 1. Domínios de desenvolvimento – objectivos

3 Obtenção do *p*GML

3.1 Engenharia reversa sobre o GML

O modelo geográfico proposto pelo OGC (`Open Geospatial Consortium`) apresenta-se sob a forma de uma especificação abstracta, o GML (*Geography Markup Language*), no qual se define uma entidade geográfica como uma "*abstracção de um fenómeno do mundo real; é uma entidade geográfica se estiver associada a uma localização relativa à Terra*". Desta forma, o mundo real pode ser representado através de um conjunto de entidades. O estado de cada entidade é definido por um conjunto de propriedades, as quais se definem por um triplo – nome, tipo e valor. A definição do tipo de entidade restringe-a quanto ao número de propriedades e tipos de propriedades que uma entidade desse tipo pode ter. O OGC indica também que as entidades geográficas se caracterizam por possuírem propriedades geométricas. E acrescenta que um conjunto de entidades pode ser visto como uma entidade em si só; por isso, uma colecção de entidades tem um tipo e propriedades particulares para além das que contém nas entidades contidas.

A W3C[14] revela que os XML *Schemas* expressam um vocabulário comum que permite as máquinas interpretarem regras e normas humanas. O OGC[3], ao especificar o GML através de XML *Schemas*, faculta a interpretação automática da norma que define, por exemplo através da utilização do mecanismo de transformação associado ao XML, XSLT (*eXtensible Stylesheet Language for Transformations*)[6][13], uma vez que um XML *Schema* é também ele próprio um documento XML.

O primeiro passo, do processo de engenharia reversa, traduziu-se na definição de uma função de transformação, um documento XSLT, no qual se aproxima a definição dos *feature.xsd*, *geometry.xsd*, *xlinks.xsd* que compõem o GML, numa primeira definição dos respectivos tipos em VDM-SL.

3.2 Obtenção de um subconjunto preciso do GML 2.12

Obtida a definição dos tipos em VDM-SL impunha-se a necessidade de analisar a sua definição formal e com isso investigar a possibilidade de obter um modelo compatível com o anterior, mais simplificado e, portanto, com um grau de abstracção superior, o *p*GML. Para o conseguir procedeu-se ao refinamento da especificação obtida, no passo anterior, pela ordem *natural* que o GML induz, ou seja, partindo da noção abstracta de entidade e da sua caracterização, até obter as suas propriedades geométricas e a respectiva definição de tipos (ou classes) dessas propriedades.

O resultado dessa análise traduz-se numa especificação em VDM-SL que aqui se apresenta em XML, após automática conversão da definição de tipos VDM-SL para XSD. O processo de construção do *p*GML, a sua sintaxe, semântica e o mecanismo de extracção do XSD e XML a partir de uma especificação VDM-SL, podem ser aprofundados em [4].

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:gml="http://www.opengis.net/gml" xmlns:xs="http://www.w3.org/2001/XMLSchema"
 elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:import namespace="http://www.opengis.net/gml" schemaLocation="feature.xsd"/>
<xs:import namespace="http://www.w3.org/1999/xlink" schemaLocation="xlinks.xsd"/>
<xs:complexType name="AbstractFeatureType">
   <xs:sequence><xs:element name="d" type="description" minOccurs="0" maxOccurs="1"/>
    <xs:element name="n" type="name" minOccurs="0" maxOccurs="1"/>
    <xs:element name="bb" type="boundedBy" minOccurs="0" maxOccurs="1"/>
    <xs:element name="fid" type="ID" minOccurs="0" maxOccurs="1"/></xs:sequence>
</xs:complexType>
<xs:complexType name="AbstractFeatureCollectionType">
   <xs:sequence><xs:element name="d" type="description" minOccurs="0" maxOccurs="1"/>
    <xs:element name="n" type="name" minOccurs="0" maxOccurs="1"/>
    <xs:element name="bb" type="boundedBy"/>
    <xs:element name="fid" type="ID" minOccurs="0" maxOccurs="1"/></xs:sequence>
</xs:complexType>
<xs:complexType name="PointPropertyType">
 <xs:choice><xs:element name="selector0" type="Point"/>
  <xs:element name="selector1" type="LinkPoint"/></xs:choice>
</xs:complexType>
<xs:complexType name="Point">
   <xs:sequence><xs:element name="gid" type="ID" minOccurs="0" maxOccurs="1"/>
    <xs:element name="srsName" type="anyURI" minOccurs="0" maxOccurs="1"/>
    <xs:element name="p" type="coord"/></xs:sequence>
</xs:complexType>
<xs:complexType name="PolygonPropertyType">
 <xs:choice><xs:element name="selector0" type="Polygon"/>
  <xs:element name="selector1" type="LinkPolygon"/></xs:choice>
</xs:complexType>
<xs:complexType name="Polygon">
   <xs:sequence><xs:element name="gid" type="ID" minOccurs="0" maxOccurs="1"/>
    <xs:element name="srsName" type="anyURI" minOccurs="0" maxOccurs="1"/>
    <xs:element name="outerBoundaryIs" type="LinearRing"/>
    <xs:element name="innerBoundaryIs" type="LinearRing" minOccurs="0"
        maxOccurs="unbounded" /></xs:sequence>
</xs:complexType>
<xs:complexType name="LinearRing">
 <xs:sequence><xs:element name="selector" type="coord"  minOccurs="4"
    maxOccurs="unbounded"/></xs:sequence>
</xs:complexType>
<xs:complexType name="LineStringPropertyType">
 <xs:choice><xs:element name="selector0" type="LineString"/>
  <xs:element name="selector1" type="LinkLineString"/></xs:choice>
</xs:complexType>
<xs:complexType name="LineString">
   <xs:sequence><xs:element name="gid" type="ID" minOccurs="0" maxOccurs="1"/>
```

```
    <xs:element name="srsName" type="anyURI" minOccurs="0" maxOccurs="1"/>
    <xs:element name="l" type="coord"  minOccurs="2"  maxOccurs="unbounded" />
   </xs:sequence>
</xs:complexType>
<xs:complexType name="MultiPointPropertyType">
 <xs:choice><xs:element name="selector0" type="MultiPoint"/>
  <xs:element name="selector1" type="LinkMultiPoint"/></xs:choice>
</xs:complexType>
<xs:complexType name="MultiPoint">
   <xs:sequence><xs:element name="gid" type="ID" minOccurs="0" maxOccurs="1"/>
    <xs:element name="srsName" type="anyURI"/>
    <xs:element name="mp" type="Point" minOccurs="0" maxOccurs="unbounded" />
   </xs:sequence>
</xs:complexType>
<xs:complexType name="MultiLineStringPropertyType">
 <xs:choice><xs:element name="selector0" type="MultiLineString"/>
  <xs:element name="selector1" type="LinkMultiLineString"/></xs:choice>
</xs:complexType>
<xs:complexType name="MultiLineString">
   <xs:sequence>
    <xs:element name="gid" type="ID" minOccurs="0" maxOccurs="1"/>
    <xs:element name="srsName" type="anyURI"/>
    <xs:element name="mls" type="LineString" minOccurs="0"
        maxOccurs="unbounded" /></xs:sequence>
</xs:complexType>
<xs:complexType name="MultiPolygonPropertyType">
 <xs:choice>
  <xs:element name="selector0" type="MultiPolygon"/>
  <xs:element name="selector1" type="LinkMultiPolygon"/>
 </xs:choice>
</xs:complexType>
<xs:complexType name="MultiPolygon">
   <xs:sequence><xs:element name="gid" type="ID" minOccurs="0" maxOccurs="1"/>
    <xs:element name="srsName" type="anyURI"/>
    <xs:element name="mpol" type="Polygon" minOccurs="0" maxOccurs="unbounded" />
   </xs:sequence>
</xs:complexType>
<xs:complexType name="MultiGeometryPropertyType">
 <xs:choice><xs:element name="selector0" type="GeometryCollection"/>
  <xs:element name="selector1" type="LinkMultiGeometry"/></xs:choice>
</xs:complexType>
<xs:complexType name="GeometryCollection">
   <xs:sequence><xs:element name="gid" type="ID" minOccurs="0" maxOccurs="1"/>
    <xs:element name="srsName" type="anyURI"/>
    <xs:element name="collection" type="MultiGeometry" minOccurs="0"
        maxOccurs="unbounded" /></xs:sequence>
</xs:complexType>
<xs:complexType name="MultiGeometry">
 <xs:choice><xs:element name="selector0" type="Point"/>
  <xs:element name="selector1" type="LineString"/>
  <xs:element name="selector2" type="Polygon"/>
  <xs:element name="selector3" type="MultiPoint"/>
  <xs:element name="selector4" type="MultiLineString"/>
  <xs:element name="selector5" type="MultiPolygon"/>
  <xs:element name="selector6" type="GeometryCollection"/></xs:choice>
</xs:complexType>
<xs:element name="LinkPoint" type="AssociationAttributeGroup"/>
<xs:element name="LinkPolygon" type="AssociationAttributeGroup"/>
<xs:element name="LinkLineString" type="AssociationAttributeGroup"/>
<xs:element name="LinkMultiPoint" type="AssociationAttributeGroup"/>
<xs:element name="LinkMultiLineString" type="AssociationAttributeGroup"/>
<xs:element name="LinkMultiPolygon" type="AssociationAttributeGroup"/>
<xs:element name="LinkMultiGeometry" type="AssociationAttributeGroup"/>
<xs:complexType name="AssociationAttributeGroup">
   <xs:sequence><xs:element name="xlink" type="simpleLink"/>
    <xs:element name="rs" type="remoteSchema" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="coord">
   <xs:sequence><xs:element name="X" type="xs:long"/>
    <xs:element name="Y" type="xs:long" minOccurs="0" maxOccurs="1"/>
    <xs:element name="Z" type="xs:long" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>


 </xs:complexType> <xs:simpleType name="anyURI">
 <xs:restriction base="string"/>
</xs:simpleType>
<xs:simpleType name="remoteSchema">
 <xs:restriction base="anyURI"/>
</xs:simpleType>
<xs:simpleType name="description">
 <xs:restriction base="string"/>
</xs:simpleType>
<xs:simpleType name="name">
 <xs:restriction base="string"/>
</xs:simpleType>
<xs:simpleType name="ID">
 <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="string">
 <xs:restriction base="xs:string"/>
</xs:simpleType>
<xs:simpleType name="href">
 <xs:restriction base="anyURI"/>
</xs:simpleType>
<xs:simpleType name="role">
 <xs:restriction base="anyURI"/>
</xs:simpleType>
<xs:simpleType name="arcrole">
 <xs:restriction base="anyURI"/>
</xs:simpleType>
<xs:simpleType name="title">
```

```
 <xs:restriction base="string"/>
</xs:simpleType>
<xs:simpleType name="show">
 <xs:restriction base="xs:string">
  <xs:enumeration value="new"/>
  <xs:enumeration value="replace"/>
  <xs:enumeration value="embed"/>
  <xs:enumeration value="other"/>
  <xs:enumeration value="none"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="actuate">
 <xs:restriction base="xs:string">
  <xs:enumeration value="onLoad"/>
  <xs:enumeration value="onRequest"/>
  <xs:enumeration value="other"/>
  <xs:enumeration value="none"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="label">
 <xs:restriction base="string"/>
</xs:simpleType>
<xs:simpleType name="NullType">
 <xs:restriction base="xs:string">
  <xs:enumeration value="inapplicable"/>
  <xs:enumeration value="unknown"/>
  <xs:enumeration value="unavailable"/>
  <xs:enumeration value="missing"/>
 </xs:restriction>
</xs:simpleType>


<xs:complexType name="simpleLink">
   <xs:sequence><xs:element name="type" type="xs:string" fixed="simple"/>
    <xs:element name="attr2" type="href" minOccurs="0" maxOccurs="1"/>
    <xs:element name="attr3" type="role" minOccurs="0" maxOccurs="1"/>
    <xs:element name="attr4" type="arcrole" minOccurs="0" maxOccurs="1"/>
    <xs:element name="attr5" type="title" minOccurs="0" maxOccurs="1"/>
    <xs:element name="attr6" type="show" minOccurs="0" maxOccurs="1"/>
    <xs:element name="attr7" type="actuate" minOccurs="0" maxOccurs="1"/>
   </xs:sequence>
</xs:complexType>
<xs:complexType name="boundedBy">
 <xs:choice><xs:element name="selector0" type="Box"/>
  <xs:element name="selector1" type="NullType"/></xs:choice>
</xs:complexType>
<xs:complexType name="Box">
   <xs:sequence><xs:element name="gid" type="ID" minOccurs="0" maxOccurs="1"/>
    <xs:element name="srsName" type="anyURI" minOccurs="0" maxOccurs="1"/>
    <xs:element name="b" type="coord" minOccurs="0" maxOccurs="unbounded" />
   </xs:sequence>
</xs:complexType>
<xs:complexType name="featureMember">
   <xs:sequence><xs:element name="feature" type="AbstractFeatureType"/>
    <xs:element name="geoproperties" type="GeometryProperty"
        minOccurs="0" maxOccurs="unbounded" /></xs:sequence>
</xs:complexType>
<xs:complexType name="GeometryProperty">
 <xs:choice>
  <xs:element name="selector0" type="PointPropertyType"/>
  <xs:element name="selector1" type="LineStringPropertyType"/>
  <xs:element name="selector2" type="PolygonPropertyType"/>
  <xs:element name="selector3" type="MultiPointPropertyType"/>
  <xs:element name="selector4" type="MultiLineStringPropertyType"/>
  <xs:element name="selector5" type="MultiPolygonPropertyType"/>
  <xs:element name="selector6" type="MultiGeometryPropertyType"/>
 </xs:choice>
</xs:complexType>
</xs:schema>
```

4 Resultados

Do estudo e do trabalho realizado advém um conjunto de resultados que se podem dividir em duas categorias. O primeiro resultado é de âmbito mais teórico e visava comprovar a capacidade expressiva do *p*GML, ao passo que o segundo conjunto de resultados permite ao engenheiro de *software* utilizar a *framework* construída, para desenvolver modelos específicos sobre o *p*GML da mesma forma que utilizaria uma outra metodologia que lhe permitisse cumprir as fases da estratégia de desenvolvimento de *software* GIS.

4.1 Resultados da obtenção do *p*GML

A obtenção do *p*GML em VDM-SL ganha valor por através dela ser possível cumprir a estratégia de desenvolvimento de *software* na qual a representação do modelo é capaz de ser derivada num documento GML, o que completa o ciclo:
Assim:

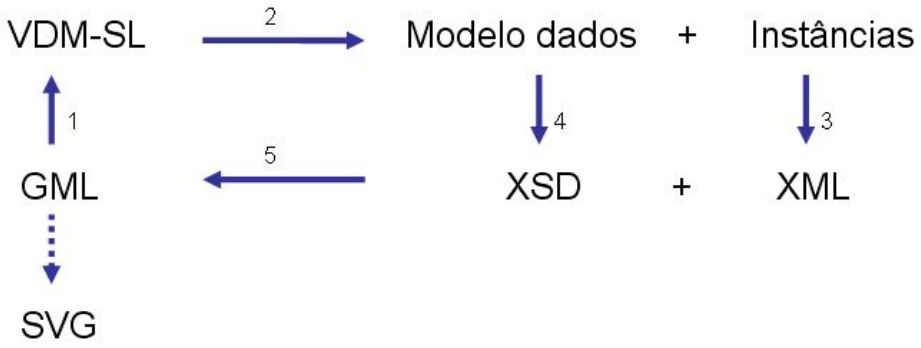

Figura 2. Ciclo: do GML ao GML

1. por engenharia reversa obteve-se um modelo preciso do GML em VDM-SL;
2. capaz de reter um modelo de dados e instâncias;
3. as instâncias são mapeadas em XML;
4. o modelo de dados permite definir o XSD;
5. o que nos retorna de novo ao GML.

Ganhou-se com isto:

- a transposição para o domínio dos Métodos Formais de um modelo de GIS;
- um mecanismo rápido de modelação e prototipagem;
- uma função de derivação da implementação a partir do modelo;
- um meio de comunicação com o cliente, utilizando por exemplo uma função de conversão de GML para SVG.

4.2 Utilização do *p*GML e da *framework* desenvolvida

O engenheiro de *software* passa a ter ao seu dispor um conjunto de facilidades que lhe permitem registar, sob forma de um modelo, os requisitos impostos para o seu Sistema de Informação Geográfica, capaz de ser animado (ie. utilizado como protótipo) de forma a obter uma correcta especificação dos requisitos. Deverá ainda utilizá-lo como *proof of concept* e meio de comunicação com o cliente. A obtenção do modelo final é obtida não de forma *ad hoc*, na qual vale ou não a experiência do engenheiro de *software*, mas por derivação passando da *abstract syntax* neste caso representada em VDM-SL para *concrete syntax*, neste caso o GML. Cada *função* de transformação (numa outra *concrete syntax*) corresponderá a uma nova implementação do mesmo modelo.

A utilização desta *framework* pode ser mais facilmente compreendida através de um pequeno *running example*:

> ”Uma instituição com várias unidades, projectos e edifícios pretende desenvolver um sistema de informação com os seus colaboradores, unidades, infra-estruturas e equipamentos, entre eles a própria localização física do posto de trabalho, contactos...”

O engenheiro de *software* irá utilizar a definição *p*GML da mesma forma que utilizaria o GML como base sobre a qual definiria o seu XSD específico para a sua representação aplicacional. Assim a definição do seu modelo deverá começar por importar todas as definições do *p*GML.

Sendo a sintaxe do VDM-SL bastante intuitiva apresenta-se de imediato o modelo de dados simplificado, ie. apenas com a indicação dos invariantes geométricos ou definidos nos requisitos.

```
module firststep
    imports
        from pgml all
    exports all
definitions
    EdificioType :: feature : pgml`AbstractFeatureCollectionType
                    Computadores : [pgml`MultiPointPropertyType]
                    Mobiliario : [pgml`MultiPolygonPropertyType]
                    Salas : SalaType-set
    inv mk-EdificioType (f, c, m, s) ≜
        EdiContainSala (s, f.bb) ∧
        EdiContainCompu (c, f.bb) ∧
        EdiContainMob (m, f.bb);
```

Onde se define que um edifício é um tipo de entidades que se traduz numa extensão a um *AbstractFeatureCollectionType*, extensão essa composta por três sub-entidades: *Computadores*, *Mobiliario* e um conjunto de *Salas*. Tem ainda um invariante associado, que irá garantir a conformidade geométrica deste tipo e das suas sub-entidades.

Por sua vez, *SalaType* define-se como:

```
    SalaType :: feature : pgml`AbstractFeatureType
                extentOf : pgml`PolygonPropertyType
                Pessoa : PessoaType-set
    inv sala ≜ card (sala.Pessoa) ≤ 10;
```

onde *SalaType* é uma entidade que estende *AbstractFeature*; é composta pela informação geométrica da sua área externa e da informação dos seus colaboradores. A título de exemplo, é imposto um invariante que limita a 10 o número máximo de colaboradores por sala.

Por fim, *PessoaType* é uma entidade com um atributo geométrico, a sua *location*, e herda, de acordo com o modelo, os atributos de *AbstractFeatureType* e tem ainda associada a informação da *idade* do colaborador:

$$
\begin{array}{l}
PessoaType :: feature : pgml`AbstractFeatureType \\
\qquad\qquad\quad location : pgml`PointPropertyType \\
\qquad\qquad\quad idade : \mathbb{N}
\end{array}
$$

A especificação do invariante de `EdificioType` é composta pela conjunção de três condições, que impõem a conformidade geométrica dos elementos contidos no edifício.

$$
\begin{array}{l}
EdiContainSala : SalaType\text{-set} \times pgml`Box \to \mathbb{B} \\
EdiContainSala\,(s, b) \triangleq \\
\quad (\forall\, sala \in s \cdot EdiContainSala2\,(sala.extentOf, b)); \\
EdiContainSala2 : pgml`Polygon \times pgml`Box \to \mathbb{B} \\
EdiContainSala2\,(pol, b) \triangleq \\
\quad (\forall\, crd \in \mathsf{elems}\ (pol.outerBoundaryIs) \cdot EdiContainSala3\,(crd, b)); \\
EdiContainSala3 : pgml`coord \times pgml`Box \to \mathbb{B} \\
EdiContainSala3\,(crd, b) \triangleq \\
\quad (crd.X \geq (\mathsf{hd}\ (b.b)).X) \wedge \\
\quad (crd.X \leq (\mathsf{hd}\ (\mathsf{tl}\ (b.b))).X) \wedge \\
\quad (crd.Y \geq (\mathsf{hd}\ (b.b)).Y) \wedge \\
\quad (crd.Y \leq (\mathsf{hd}\ (\mathsf{tl}\ (b.b))).Y) \wedge \\
\quad (crd.Z \geq (\mathsf{hd}\ (b.b)).Z) \wedge \\
\quad (crd.Z \leq (\mathsf{hd}\ (\mathsf{tl}\ (b.b))).Z); \\
EdiContainCompu : pgml`MultiPointPropertyType \times pgml`Box \to \mathbb{B} \\
EdiContainCompu\,(c, b) \triangleq \\
\quad (\forall\, e \in c.mp \cdot EdiContainCompu2\,(e, b)); \\
EdiContainCompu2 : pgml`Point \times pgml`Box \to \mathbb{B} \\
EdiContainCompu2\,(p, b) \triangleq \\
\quad (p.p.X \geq (\mathsf{hd}\ (b.b)).X) \wedge \\
\quad (p.p.X \leq (\mathsf{hd}\ (\mathsf{tl}\ (b.b))).X) \wedge \\
\quad (p.p.Y \geq (\mathsf{hd}\ (b.b)).Y) \wedge \\
\quad (p.p.Y \leq (\mathsf{hd}\ (\mathsf{tl}\ (b.b))).Y) \wedge \\
\quad (p.p.Z \geq (\mathsf{hd}\ (b.b)).Z) \wedge \\
\quad (p.p.Z \leq (\mathsf{hd}\ (\mathsf{tl}\ (b.b))).Z); \\
EdiContainMob : pgml`MultiPolygonPropertyType \times pgml`Box \to \mathbb{B} \\
EdiContainMob\,(m, b) \triangleq \\
\quad (\forall\, e \in m.mpol \cdot EdiContainMob2\,(e, b)); \\
EdiContainMob2 : pgml`Polygon \times pgml`Box \to \mathbb{B} \\
EdiContainMob2\,(pol, b) \triangleq \\
\quad EdiContainSala2\,(pol, b) \\
\mathsf{end}\ \textit{firststep}
\end{array}
$$

Construído o modelo o engenheiro de *software* pode utilizar o mecanismo de geração do XML e do XSD a partir do módulo VDM-SL[4][9][15] e com isso obter uma

implementação completa e automaticamente derivada a partir do modelo. A imagem seguinte ilustra uma instância XML de *EdificioType*.

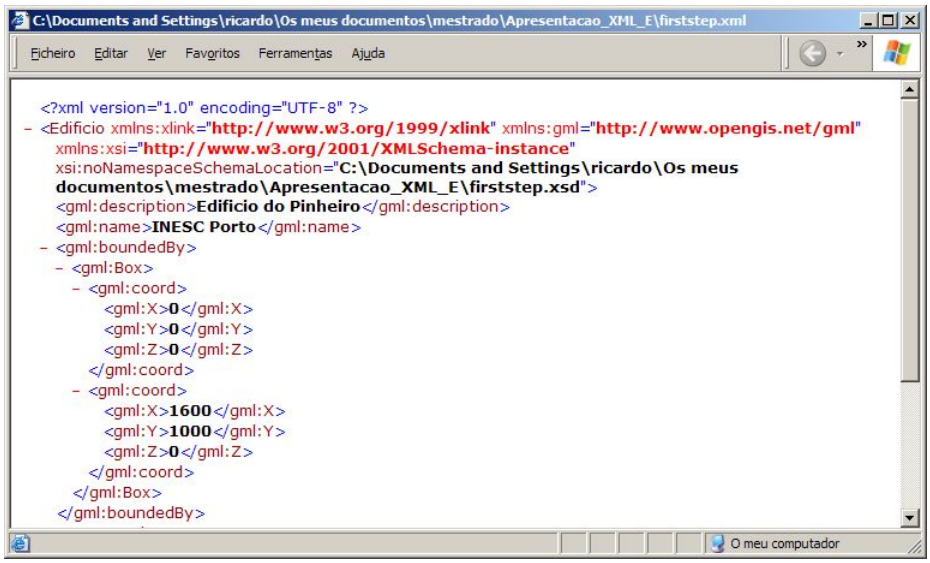

Figura 3. XML gerado a partir da *abstract syntax*

Através da ferramenta definida em [11] pode-se obter uma apresentação em SVG o que pode facilitar a comunicação com o cliente, como se ilustra na figura 4.

5 Conclusões e trabalho futuro

5.1 Conclusões

O GML, como modelo de armazenamento e como veículo de troca de informação geográfica é robusto e coerente. Estabelece um *standard* e um marco importante para a interoperabilidade que tanto em voga está. Contudo, como modelo de representação de um sistema de informação geográfica, não é ainda a mais valia que se procurava, no sentido em que apresenta algumas limitações, nomeadamente, por não introduzir estruturas algébricas de alto nível capazes de abstrair as entidades gráficas de baixo nível.

A sua forma de modelação *object-oriented* configura uma simplicidade e uma capacidade de derivação que, a ser estendida para outro nível poderá, então, garantir a conformidade semântica entre o elemento representado e a sua definição geométrica, e com isso a forma de como os elementos de representação do mundo real se articulam, para captarem, fielmente, as características e invariantes das entidades reais que representam. De facto o GML[2] deve ser encarado como uma plataforma genérica e

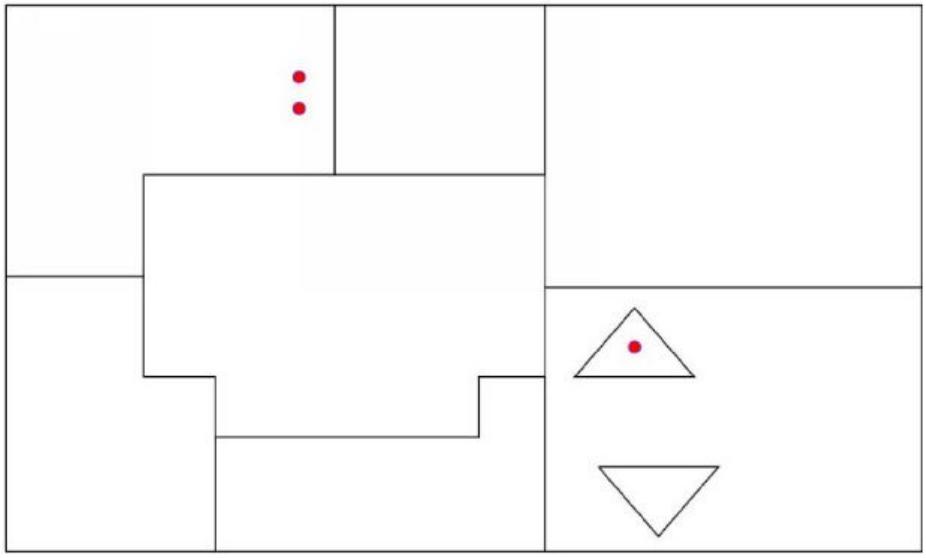

Figura 4. Exemplo do SVG obtido

rica que define apenas um patamar de abstracção para o armazenamento e partilha de informação geográfica.

O trabalho sucintamente descrito permitiu obter uma definição formal de um subconjunto do GML, traduzido por uma modelação sob o eixo dos dados. Uma especificação formal neste domínio tão vasto é apenas um primeiro passo e um exemplo sobre o qual se podem derivar novos modelos e implementações, mais ou menos ricas, consoante a necessidade e a capacidade de computação disponível. Conseguiu-se importar para o domínio dos Sistemas de Informação Geográfica o rigor, simplicidade, capacidade de raciocínio e prototipagem dos métodos formais e dualmente alargar o domínio dos métodos formais.

A capacidade de obter uma representação de um micro Sistema de Informação Geográfica, que respeite o GML, a partir de um modelo formal indica que é possível obter a mesma representação para qualquer implementação, bastando para isso mudar a função de derivação. Neste sentido é também possível derivar o *p*GML para paradigmas específicos utilizados em *softwares* comerciais.

5.2 Trabalho futuro

Na perspectiva histórica, sob o ponto de vista da interoperabilidade, ao olhar o XML, ou especificamente o GML, como uma espécie de denominador comum para a intercomunicação de diversos sistemas não compatíveis e com formatos proprietários, é compreensível que se tenha deixado o ónus da conformidade geométrica recair sobre

esses sistemas. Contudo a riqueza semântica do VDM-SL, no que respeita aos invariantes geométricos (geográficos), necessita também de ser expressa. De acordo com [8] o XSCL (*XML Constraint Specification Language*) é o candidato *natural* a representar os invariantes semânticos (que resultam da necessidade de representação de elementos reais, os quais obedecem a constrangimentos reais[10]), sobre outras abordagens como o *Schematron* e *XML-Schema*. O XCSL é ainda ele próprio um formato XML[7].

Referências

[1] J. Backus. Can programming be liberated from the von Neumann style? a functional style and its algebra of programs. *CACM*, 21(8):613–639, August 1978.

[2] Open Gis Consortium. Gml implementation specification. Technical Report 02-023r4.

[3] Open GIS Consortium. *OpenGIS Geography Markup Language Implementation Specification*. OGC, 2002.

[4] Mário Ricardo de Novais Henriques. pgml-estudo de um subconjunto preciso do gml2.12. Technical report, Univ. Minho, 2004.

[5] The VDM Tool Group. Vdmtools - the ifad vdm-sl language. Technical report, IFAD, Forskerparken 10, DK-5230 Odense M, Denmark, 2000.

[6] Kal Ahmed; Sudhir Ancha; Andrei Cioroianu; Jay Cousins; JereMy Crosbie; John Davies; Kyle Gabhart; Steve Gould; Ramnivas Laddad; Sing Li; Brendan Macmillan; Daniel Rivers-Moore; Judy Skubal; Karli Watson; Scott Williams; James Hart. *Professional Java XML*. WROX Press Inc., 2001.

[7] Marta Jacinto; Giovani Librelotto; José Ramalho; Pedro Henriques. Xcsl tutorial. Technical report, Universidade do Minho, 2002.

[8] Marta Jacinto; Giovani Librelotto; José Ramalho; Pedro Henriques. Xcsl: Xml constraint specification language. Technical report, Universidade do Minho, 2003.

[9] IFAD. Vdmtools - the dynamic link facility. Technical report, IFAD, Forskerparken 10, DK-5230 Odense M, Denmark, 2000.

[10] Martien Molenaar. *An Introduction to the Theory of Spatial Object Modelling for GIS*. Taylor and Francis, 1998.

[11] Anabela Soares Pinto Monteiro. Servidor de mapas vectoriais. relatório do projecto. Technical report, 2002.

[12] J. N. Oliveira. CAD Tool Extension for Formal Building Description Language . *Advances in Engineering Software http://www.elsevier.nl/inca/publications/store/4/2/2/9/1/1/index.htt*, 29(7-9):571–586, 1998.

[13] Pedro Rangel Henriques; José Carlos Ramalho. *XML & XSD - da teoria à prática*. FCA, 2002.

[14] W3C. W3c architecture domain, xml schema. (http://www.w3c.org/XML/Schema).

[15] Larry Wall and Randal L. Schwartz. *Programming Perl*. O'Reilly & Associates, Inc, 1992.