Utilizando a GML na identificação de candidatos a padrão de análise para BDG

GUILLERMO NUDELMAN HESS[1]
CIRANO IOCHPE[1]
[1]UFRGS—Universidade Federal do Rio Grande do Sul, Caixa Postal 15064, Porto Alegre, RS, Brasil
{hess,ciochpe}@inf.ufrgs.br

Abstract. This paper describes the development of a software architecture for the pre-processing of data in the Knowledge Discovery in Database (KDD) process, specifically for conceptual modeling of Geographic Databases. The goal is to create a mechanism to (semi)-automate the conversion of conceptual schemas based on different data models into a single canonical format, syntactically and semantically, and from that format to one accepted by the data mining softwares. Rules are presented for the syntactic conversion from UML-GeoFrame to the Geographic Markup Language (GML) and from GML to FDE.

1 Introdução

Com o aumento da utilização dos Sistemas de Informação Geográfica (SIG), nos últimos anos, a fase de modelagem conceitual dos Bancos de Dados Geográficos (BDG) tornou-se uma atividade extremamente importante. Ocorre, entretanto, que cada software de SIG apresenta um modelo de dados próprio, focado muito mais na etapa lógica do projeto de banco de dados [15].

Diversos modelos conceituais para projeto de BDG têm sido propostos, com o objetivo de tornar a modelagem independente de plataforma de implementação. Dentre eles, pode-se citar o UML-GeoFrame [7, 12], o MADS [10], o OMT-G [2], o padrão canadense SAIF [14] e o GeoOOA [6]. Em grande parte, estes modelos se equivalem, e um estudo comparativo entre eles aparece em [1].

O emprego da modelagem conceitual permite não somente a independência do software no qual o banco de dados será implementado e a documentação do projeto, como também a reutilização de modelo, ou de parte dele, diversas vezes. Esta reutilização é especialmente interessante no que diz respeito aos BDGs, uma vez que sua modelagem é bastante complexa, e parte da realidade geográfica modelada se repete para diferentes aplicações. Neste sentido, a utilização de padrões de análise [5], que apresentam a essência da modelagem conceitual de uma solução para um problema recorrente, em um contexto específico [8], torna-se útil.

Para dar suporte ao reconhecimento de candidatos a padrão de análise de forma automatizada, o processo de descoberta de conhecimento em banco de dados (KDD) [4] pode ser aplicado. Este processo compreende diversas etapas, conforme a Figura 1. A principal etapa é a mineração de dados (MD). Contudo, para se chegar a ela, é necessário o pré-processamento e a preparação dos dados de entrada.

Em [15] foram investigadas e desenvolvidas técnicas de MD e pós-processamento para a inferência de candidatos a padrão de análise, partindo-se de esquemas conceituais de BDG. Contudo, como aquele trabalho não tratou da preparação de esquemas reais de BDG para mineração, utilizou-se, como entrada da MD, esquemas criados artificialmente. Não foi, portanto, possível (e nem era este o objetivo do trabalho) encontrar candidatos a padrão de análise baseados em esquemas reais de BDG.

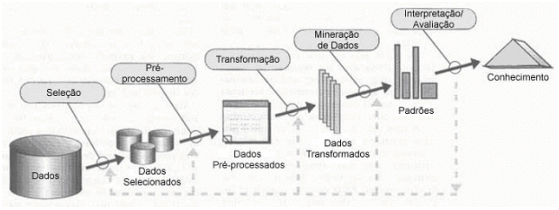

Figura 1 - O processo de KDD [4]

Para que se possa minerar diversos esquemas conceituais de BDG, de diferentes organizações e com diferentes finalidades, é necessário que eles estejam no mesmo formato, o que não ocorre com os modelos conceituais de BDG, uma vez que não existe um padrão de modelagem universalmente aceito. Para minerar candidatos reais a padrão de análise para BDG, torna-se necessário, antes, reunir esquemas diversos, cada um baseado em um modelo conceitual diferente, e convertê-los para algum modelo comum [1]. A partir daí, tais esquemas devem ser armazenados em um formato que facilite seu processamento por ferramentas de MD que produzem regras associativas [15].

Este artigo apresenta uma solução para a questão da preparação de esquemas de BDG para a mineração. A idéia central é criar um mecanismo no qual esquemas conceituais desenvolvidos com base nos diferentes modelos citados sejam convertidos para um modelo de dados canônico, independente de plataforma. É desejável também que este modelo de dados seja reconhecido como formato de intercâmbio e

armazenamento de informações geográficas. Uma vez tendo o esquema conceitual descrito em um modelo de dados único, este é codificado em um formato aceito como entrada para ferramentas de MD.

Para uma correta preparação dos dados para mineração, esta integração de esquemas deve ocorrer tanto em nível sintático quanto em nível semântico. O primeiro diz respeito à equivalência em termos dos construtores de cada modelo. Um estudo para a unificação dos conceitos existentes nos referidos modelos foi iniciado em [1], no qual é apresentado o conjunto união de construtores.

O nível semântico da integração abrange a questão da unificação da nomenclatura utilizada para representar os fenômenos da realidade a serem modelados, e os relacionamentos entre eles. Neste sentido, é necessário criar uma estrutura de organização do conhecimento, tal como um vocabulário controlado, uma taxonomia, um thesaurus ou uma ontologia [11]. A proposta aqui apresentada baseia-se em uma ontologia geográfica, com o objetivo de integrar semanticamente os modelos, identificando sinônimos e homônimos para os conceitos a serem representados.

Neste artigo, é abordada a questão sintática da unificação de modelos, e é proposta uma arquitetura de software, para (semi-) automatizar a preparação de esquemas de BDG para mineração. Com base na arquitetura proposta, um conjunto de ferramentas já foi implementado, o qual traduz esquemas GeoFrame [7] em esquemas GML [9] e posteriormente em FDE [15] para mineração. No futuro, estas ferramentas deverão ser estendidas para suportar a conversão de esquemas de outros modelos.

O restante do artigo está organizado como segue. A Seção 2 apresenta a arquitetura genérica da ferramenta de software que implementa o mecanismo de unificação de esquemas. A Seção 3 descreve, sucintamente, o framework conceitual GeoFrame. A Seção 4 apresenta um resumo da linguagem de marcação GML. A Seção 5 detalha a arquitetura e implementação do sistema, desde a criação do diagrama de classes baseado no GeoFrame, até a geração dos dados de entrada para mineração. A Seção 6 apresenta as conclusões e trabalhos futuros.

2 O mecanismo de unificação de modelos

Na etapa de mineração de dados do processo de KDD em banco de dados, com busca à identificação de candidatos a padrão de análise para BDG, um dos problemas que se apresentou para Silva [15] foi o fato de não existirem esquemas conceituais reais de BDG em um modelo de dados que pudesse servir de entrada para as ferramentas de MD. Como esta questão não era o ponto central do trabalho, não foi enfrentada.

O presente artigo visa a complementar o trabalho apresentado em [15], preparando os dados a serem utilizados no processo de MD. Para possibilitar a mineração de esquemas conceituais baseados em diversos modelos, faz-se necessário o desenvolvimento de um mecanismo de unificação desses modelos. Esta integração deve ocorrer tanto em nível conceitual quanto no nível semântico, a fim de não haver ambigüidades de interpretação e tampouco redundância de conceitos.

Neste sentido, a Figura 2 apresenta uma arquitetura genérica de tradução de esquemas conceituais, independente de modelo de dados utilizado. Conforme ilustrado, um esquema conceitual é convertido, primeiramente, para um arquivo em formato canônico sintático (AFCS), ou seja, somente no nível sintático. De acordo com o modelo de dados no qual o esquema conceitual está baseado, um determinado conjunto de regras é aplicado. A segunda etapa do processo consiste em passar o AFCS por uma ontologia, de modo a garantir o nível semântico da preparação de dados. O resultado é um arquivo no formato canônico sintático e semântico (AFCSS). A última etapa da preparação dos dados para mineração consiste em transformar o arquivo AFCSS no formato de dados FDE [15], suportado por ferramentas de MD.

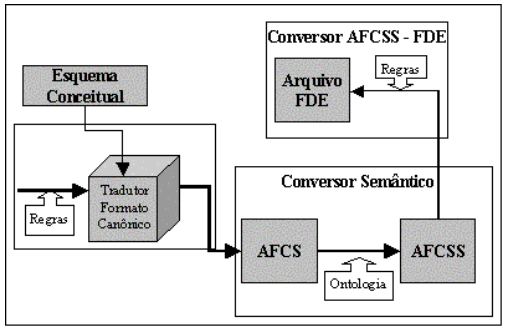

Figura 2 - Arquitetura genérica do mecanismo de preparação de dados

Para uma primeira implementação e verificação da eficácia e aplicabilidade do mecanismo, optou-se pela conversão de um esquema conceitual baseado no *framework* GeoFrame para a GML.

Mesmo sabendo que a GML não é capaz de representar todos os construtores dos demais modelos, ela foi adotada, neste trabalho, por apresentar um conjunto significativo de elementos de modelagem de BDG. Além disso, a GML pode ser estendida, no futuro, para suportar os construtores faltantes.

3 O *framework* UML-GeoFrame

O GeoFrame [7] é um *framework* conceitual que fornece um diagrama de classes básicas para dar suporte ao projetista na modelagem conceitual de dados geográficos para aplicações de SIG. Ele foi desenvolvido no Instituto de Informática da UFRGS, estando, atualmente, na versão 2.0, tendo também uma extensão temporal, chamada GeoFrame-T [12].

O GeoFrame é baseado no paradigma da orientação a objetos e no padrão de modelagem *Unified Modeling Language* (UML). As classes do *framework*, na sua versão 2 [12] são descritas a seguir.

- Objeto Não-Geográfico: Objetos convencionais, ou seja, aqueles que não são georeferenciados. Eles não possuem referência em relação à sua posição geográfica, nem representação espacial (geometria).

- Metadado: Dados que descrevem os próprios dados, especificamente dos objetos não-geográficos.

- Geometadado: Especialização da classe Metadado, especificamente para fenômenos geográficos.

- FenômenoGeográfico: Classe que representa qualquer fenômeno de interesse da aplicação, cuja localização geográfica e representação espacial (geometria) deve ser considerada.

- CampoGeográfico: Especialização da classe FenômenoGeográfico, que generaliza os fenômenos que se enquadram na visão de campo geográfico, ou seja, apresentam uma distribuição contínua no espaço. A realidade é modelada como um todo, não sendo possível analisar individualmente cada parte.

- ObjetoGeográfico: Especialização da classe FenômenoGeográfico, que generaliza os fenômenos que se enquadram na visão de objeto, ou seja, cada objeto é modelado individualmente. Cada um deles apresenta suas características particulares, descritas por meio de seus atributos.

- ObjetoEspacial: Classe que apresenta um conjunto de construtores necessários para representação espacial de objetos geográficos. É especializada nas subclasses Ponto, Linha, Polígono e ObjetoEspComplexo.

- RepresentaçãoCampo: Classe usada para a representação espacial de campos geográficos. É especializada nas subclasses GradeCélulas, PolAdjacentes, Isolinhas, GradePontos, TIN e PontosIrregulares.

Visando deixar o diagrama mais limpo, o GeoFrame faz uso dos construtores da UML denominados estereótipos, os quais representam a geometria do objeto geográfico. Sua semântica é de substituição da associação entre uma classe que representa um fenômeno geográfico e uma classe que representa sua geometria.

4 O padrão GML

A Geographic Markup Language (GML) é uma codificação XML para transporte e armazenamento de informação geográfica, incluindo suas propriedades espaciais e não espaciais [9]. É um padrão proposto pelo consórcio OpenGIS (OGC). A especificação da GML está baseada na sintaxe XML-Schema e provê mecanismos e convenções para:

- Geração de um *framework* aberto para definição de esquemas e objetos de aplicações geoespaciais;

- Suporte à descrição de esquemas de aplicações geoespaciais específicas para um domínio;

- Suporte ao armazenamento e transporte de aplicações e conjunto de dados;

- Possibilidade de troca de informações e esquemas de aplicações geoespaciais entre organizações.

A GML está baseada na especificação feita pelo OpenGIS consortium para modelagem dos aspectos geográficos do mundo. Neste sentido, são reconhecidos fenômenos de zero, uma ou duas dimensões. Não há suporte para geometrias tridimensionais. No modelo GML, as geometrias tradicionais de zero, uma e duas dimensões são definidas num sistema de referência espacial (SRS) bidimensional, e são representados como pontos, linhas e polígonos. Ainda há representação para coleções de geometrias.

Os fenômenos do mundo real são tratados, na GML, como feições na visão de objetos. Estas *features* possuem propriedades, tanto espaciais (geométricas) quanto descritivas.

Além de ser baseada em XML-Schema, a GML 2.0 também foi concebida de forma consistente com XML *namespaces*. Assim, todo documento GML é definido com base em três esquemas para codificação da informação espacial:

- *Geometry schema* (geometry.xsd): possui o detalhamento das geometrias suportadas;

- *Feature schema* (feature.xsd): define as propriedades geográficas das *features* do modelo;

- *Xlinks* (xlinks.xsd): provê os atributos de *xlink* para a implementação das funcionalidades de ligação.

Juntos, estes três esquemas fornecem um meta-esquema, ou seja, um conjunto de classes básicas sobre as quais o usuário vai criar suas extensões, de acordo com as necessidades de sua aplicação. Quando codificadas, cada classe mo meta-esquema transforma-se em um elemento GML.

O alicerce para a capacidade da GML de expressar geometrias é o *geometry schema*. De acordo com o modelo da OGC, a GML possui os elementos correspondentes às classes geométricas *Point* (ponto), *LineString* (linha), *LinearRing* (anel de linhas), *Polygon* (polígono), *MultiPoint* (coleção de pontos), *MultiLineString (*coleção de linhas), *MultiPolygon* (coleção de polígonos) e *MultiGeometry* (múltiplas geometrias).

As geometrias definidas pela GML são usadas para a codificação dos elementos geográficos cuja representação espacial deve ser mantida, ou seja, é necessário descrever a forma e localização de determinada feição.

Na GML, o elemento básico para descrever uma feição chama-se *feature*. Uma *feature* pode estar associada a uma geometria. Assim, a geometria de uma

feição é dada através de um relacionamento. Todas as feições geográficas de uma aplicação são expressas pela instanciação da classe *AbstractFeature* ou de *AbstracFeatureCollection*.

Além de ser capaz de descrever fenômenos geográficos, a GML também tem a potencialidade de codificar os relacionamentos entre as feições. Os elementos associados podem ser declarados internamente ao relacionamento ou referenciados, utilizando *xlink* (caso sejam externos ao documento). A declaração interna, contudo, somente é possível para associações (1:1) e (1:n).

5 O sistema baseado em GML para preparação de esquemas para mineração

Uma primeira implementação do mecanismo de software proposto foi feita para suporte à conversão de esquemas conceituais de BDG baseados no *framework* GeoFrame na linguagem GML. Posteriormente, os arquivos GML são convertidos no formato de dados FDE [15], descrito na Seção 5.3. A Figura 3 ilustra a arquitetura do sistema desenvolvido, baseado no GeoFrame e em GML. Esta implementação somente leva em conta os aspectos sintáticos.

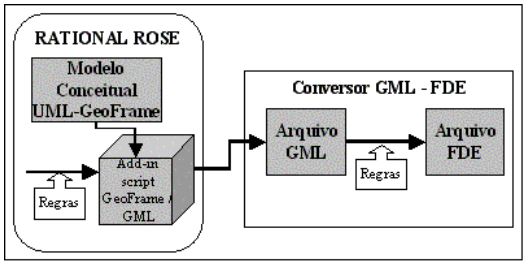

Figura 3 - Arquitetura do sistema

5.1 Ambiente Gráfico

A primeira fase do processo é a criação do modelo conceitual GeoFrame do banco de dados geográficos. Uma vez que este *framework* utiliza a notação UML, foi escolhido o ambiente Rational Rose [13] para servir de interface gráfica para o usuário, e como ambiente de conversão do esquema conceitual para GML. Esta escolha deu-se em virtude desta ferramenta CASE suportar bem todos os conceitos do GeoFrame, gerar código para os diagramas de classes, permitir programação através de *add-ins* e da licença acadêmica existente na UFRGS para utilizá-lo.

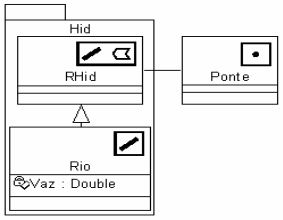

Figura 4 - Diagrama de classes GeoFrame

A Figura 4 apresenta um diagrama de classes baseado no GeoFrame e desenvolvido no Rational Rose, com tema, classes, atributos e relacionamentos.

5.2 A geração do código GML

Uma vez que a GML se baseia em XML-Schema, com extensão para encapsular aspectos geométricos e geo-espaciais dos objetos, pode-se utilizar algumas regras aplicáveis à tradução de esquemas UML para XML-Schema [3], fazendo as adaptações e extensões necessárias.

Regra 1 – Temas

Originalmente, a GML não possui um conceito semelhante ao de pacote da UML. Nos trabalhos encontrados, basicamente na Internet, que faziam o mapeamento UML para GML o construtor pacote era ignorado. Um tema (*package*) existente no modelo conceitual UML-GeoFrame é mapeado em GML como sendo um elemento, cujo atributo *name* é o próprio nome do pacote, e o atributo *type* é um novo tipo de dado, definido pelo usuário. O nome deste novo tipo de dado é o nome do pacote concatenado com *type*, e é um tipo complexo, de conteúdo complexo. Ele deve ser uma extensão da classe básica *FeatureCollection*, original da GML. Todos os componentes do diagrama de classe que estiverem contidos no tema, sejam eles classes, outros temas ou associações, serão membros dessa coleção (extensões de *FeatureMember*).

Caso exista, no modelo conceitual, uma hierarquia de pacotes, ela é mapeada para a GML através do atributo *extension base*. Isto é feito na declaração do tipo do elemento do pacote, assim como para uma classe. Uma vez que um pacote pode estar contido apenas em um, e somente um, outro construtor deste tipo, a restrição de que cada elemento pode ser de extensão a apenas um outro não é um problema.

Sendo a GML baseada em XML-Schema, é obrigatório que todo documento contenha um elemento raiz. No caso específico da GML, este elemento raiz deve ser de substituição a *FeatureCollection*. Assim, para fins de padronização de todos os documentos, é criado sempre um elemento de nome *Esquema*, de substituição a *FeatureCollection*. Este elemento será, quando for criado um XML com base neste arquivo, o elemento raiz, mas que não faz parte do modelo conceitual. Todos os demais temas estarão instanciados dentro dos limites <esquema></esquema>.

Adicionalmente, são criados dois outros elementos para cada tema mapeado. O primeiro deles é de substituição a *FeatureMember*, e o atributo *name* é formado pelo nome do tema concatenado com *Member* (por exemplo, TemaMember). O atributo *type* deste elemento é definido no mesmo arquivo, definindo uma associação. O outro elemento é de substituição a *Feature*, e o atributo *name* é formado pelo nome do tema concatenado com *Feature* (por exemplo,

TemaFeature). Este tem o atributo *type* = *"AbstractFeatureType"*.

Regra 2 – Classes

Cada classe do diagrama UML-GeoFrame é convertida para um elemento, cujo atributo *name* é o próprio nome da classe e o tipo é definido como sendo a concatenação do nome da classe com *type*. Este é tipo complexo (*complextype*), de conteúdo complexo (*complexcontent*). As classes são consideradas especializações de *AbstractFeatureType* Deste modo, herdam os atributos *fid*, *name*, *description* e *boundedby*.

No contexto deste trabalho, na definição do elemento que representa cada classe, é especificado que o atributo *SubstitutionGroup* dela é aquele elemento que representa os componentes do pacote ao qual a classe pertence. Isto é feito para poder associar uma classe ao seu tema.

Regra 3 – Atributos

Cada atributo de uma classe do diagrama UML-GeoFrame é mapeado em GML como sendo um sub-elemento do tipo complexo definido pela classe a qual o atributo pertence. O tipo do atributo mantém-se o mesmo.

Regra 4 – Associação

Qualquer tipo de associação existente num diagrama UML-GeoFrame (associações binárias, composição e agregação) é mapeada para um novo tipo complexo (*complextype*), de conteúdo complexo (*complexcontent*), com a restrição de ser uma associação de feições (*restriction base* = *"gml:AssociationTypeMember"*). Este novo tipo é formado por dois elementos, que são referências a cada um dos elementos correspondentes às classes associadas. A cardinalidade da associação é definida pelos atributos *minoccurs* e *maxoccurs*, em cada uma das referências.

Cabe salientar que os objetos referenciados não são, na realidade, os elementos que representam as classes, mas sim identificadores indicados pela concatenação do nome da classe e *Id*. Estes identificadores pertencem ao tipo *IdPropertyType*, o qual permite que o identificador seja um *ObjectIdentifier* (OID) existente no mesmo arquivo ou uma referência a um OID existente em outro arquivo de dados.

Regra 5 – Herança

Uma hierarquia de classes de um diagrama UML-GeoFrame é convertida para uma hierarquia de tipos em GML, onde cada classe do modelo conceitual é codificada como um elemento. O tipo do elemento filho é sempre uma extensão baseada no tipo do elemento pai, ou seja, as classes especializadas são declaradas como extensão da classe genérica. Assim, os atributos da classe pai passam automaticamente para as classes filhas. Como em GML cada tipo pode estender apenas baseada em um e apenas um outro tipo, herança múltipla não é permitida.

Regra 6 – Aspectos geométricos

Por se tratar de um *framework* para aplicações geográficas, o GeoFrame permite a representação das geometrias das entidades geográficas. Isto é feito através da utilização de estereótipos nas classes espaciais do modelo conceitual. Em GML, um conjunto de propriedades, associadas às formas geométricas, podem ser utilizadas. Quando mais de um tipo de geometria puder ser usado, para representar o mesmo estereótipo do diagrama de classes, fica a cargo do usuário a escolha.

Quando houver a ocorrência de múltiplas geometrias para uma determinada classe, essa característica é mapeada em GML como uma escolha (*choice*) dentre as representações, visto que, na instanciação, somente pode haver uma geometria.

5.3 O formato de dados FDE

Em [15] foi proposto um formato de dados para ser usado como entrada no processo de mineração de esquemas conceituais de BDG, com o intuito de reconhecer candidatos a padrões de análise. Este formato de dados considera que esquemas conceituais são definidos como transações no arquivo de entrada.

Cada esquema deve, na medida do possível, se decomposto em sub-esquemas, pois quanto menor o número de elementos que formam o sub-esquema, maior a chance desta estrutura se repetir em esquemas de aplicações distintas [8]. Os elementos contemplados neste trabalho são alguns dos construtores da UML presentes no UML-GeoFrame: Pacote, classe, atributo e associação binária simples.

Foram definidos dois grupos de construtores, os fortes e os fracos. Os fortes, formados por classe e pacote, são aqueles que têm significados próprios, ou seja, podem aparecer sozinhos ou associados à algum outro construtor. Já os elementos pertencentes ao grupo dos fracos, que são atributo e associação binária simples, não têm sentido soltos. Seu significado é dado por seu contexto, ou seja, somente aparecem relacionados ao(s) construtor(es) forte(s) ao(s) qual(is) pertence.

A seguir são apresentados os sub-esquemas reconhecidos pelo formato de dados proposto [15]:

- Pacote: Apresenta apenas uma instância do construtor pacote. Cada pacote do modelo conceitual origina um sub-esquema deste tipo.

- Classe: Apresenta apenas uma instância do construtor classe. Cada classe do modelo conceitual origina um sub-esquema deste tipo.

- Atributo-Classe: Apresenta apenas uma instância do construtor atributo, associado à classe a qual pertence. Apenas o nome do atributo é armazenado, seu tipo é ignorado. A representação espacial da classe também é tratada como sendo um atributo. Cada atributo de cada classe presente no modelo conceitual origina um sub-esquema deste tipo.

- Classe-Pacote: Apresenta apenas uma instância do construtor classe, associada ao pacote ao qual pertence. Cada classe do modelo conceitual que está contida num pacote origina um sub-esquema deste tipo.

- Associação: Representa as associações binárias simples, sendo que cada instância deste construtor, juntamente com as classes que ele relaciona no modelo conceitual, gera um sub-esquema deste tipo.

- Atributo-Classe-Pacote: Análogo ao sub-esquema Atributo-Classe, mas também apresenta o pacote ao qual a classe, da qual o atributo é parte, pertence.

O arquivo de entrada para mineração, contendo os sub-esquemas, deve ser plano (*flat file*). Ele é formado por vários esquemas, cada um correspondendo a uma transação. Cada transação é composta por uma marca de esquema, e por um ou mais itens de transação, os quais são os sub-esquemas gerados a partir da decomposição do esquema.

Com o objetivo de garantir a semântica de cada elemento, e de possibilitar que ao final da mineração os tipos de cada elemento sejam identificados, os dados de entrada são descritos segundo notação própria:

5.4 Regras de conversão GML para o formato de dados de entrada suportado pelas ferramentas existentes

Nem todos os elementos presentes na GML são suportados pelo formato de dados descrito acima. Apenas os elementos tema, classe, atributos, associações binárias e aspectos geográficos são contemplados. A herança, aspecto muito comum em modelagem de SIG, é desconsiderada neste mapeamento. As regras de conversão GML para este formato de dados de entrada são descritas a seguir.

Regra 7 – Temas

Cada tema (pacote) existente na GML está codificado como sendo um elemento de substituição a *FeatureCollection*, direta ou indiretamente. Se o pacote for raiz de sua hierarquia, será de substituição direta a *FeatureCollection*. No caso deste tipo de consrutor estar contido em outro tema, a substituição será indireta, ou seja, o atributo *SubstitutionGroup* apontará para o pacote que o engloba. Assim, sempre que um elemento tiver o atributo *SubstitutionGroup = "gml:_FeatureCollection"* ou a algum elemento que o possua, ele deve ser mapeado como sendo um tema no formato de dados de entrada.

Regra 8 – Classes

Cada classe existente na GML está codificada como sendo um elemento de substituição a *Feature*. Contudo, para poder relacioná-lo ao tema ao qual pertence, o elemento que representa a classe não substitui diretamente a *Feature,* e sim ao elemento que representa, genericamente, os membros do tema. Assim, sempre que o elemento possuir o atributo *SubstitutionGroup = "NometemaFeature"*, ele deve ser mapeado como sendo uma classe no formato de dados de entrada para mineração. Adicionalmente, também é criada uma entrada neste arquivo contendo *tema*: *classe*.

Regra 9 – Atributos

Cada atributo está, na GML, obrigatoriamente relacionado a uma classe. Ele é definido como sendo um elemento, e sua declaração é feita internamente à classe a qual pertence, ou seja, dentro dos limites da definição do tipo da classe <complexType></complexType>. Deste modo, sempre que for encontrada a ocorrência de um *element* dentro de um *ComplexType,* mais especificamente delimitado pelas *tags* <extension></extension> e não for uma referência a um OID, este elemento deve ser mapeado como sendo um atributo no formato de dados de entrada para mineração, associado à classe a qual pertence.

Regra 10 – Associações binárias

Associações binárias são os únicos tipos de relacionamentos suportados pelo formato de dados de entrada aqui utilizado. No arquivo GML, cada associação está identificada como sendo um tipo complexo com a restrição de ser uma *AssociationTypeMember*. Assim, sempre que um elemento possuir um tipo complexo com *restriction base = "AssociationTypeMember"*, ele será mapeado como uma associação entre as classes especificadas pelos OIDs referenciados dentro dos limites <sequence></sequence>. É importante observar que a declaração dos membros de um tema também ussa *AssociationTypeMember*, então o filtro é feito pelo texto *Member*, que, se presente, é considerado uma membro de tema, e não uma associação. A cardinalidade é ignorada.

Regra 11 – Aspectos geométricos

Os aspectos geométricos presentes na GML estão codificados como elementos, representando um atributo da classe. No mapeamento para o formato de dados de entrada para mineração esta estrutura é mantida, ou seja, a representação geométrica é tratada simplesmente como mais um atributo da classe.

Contudo, como os nomes não são iguais para representar uma mesma geometria, a Tabela 1 apresenta a geometria GML e a correspondente geometria no formato de dados de entrada.

Tabela 1 - Correspondência entre as geometrias da GML e do formato de dados de entrada

GML	Formato de dados de entrada
Location	Ponto
Position	Ponto
CenterOf	Ponto
CenterLineOf	Linha
EdgeOf	Linha
ExtentOf	Polígono
Coverage	Polígono

5.5 Execução do sistema

Para testar a correção das regras de tradução propostas, bem como da eficácia das ferramentas implementadas, foi realizado um estudo de caso. Neste teste, foi utilizado o diagrama de classes da Figura 4.

Após a execução do *script* de conversão UML-GeoFrame para GML, obteve-se um arquivo salvo com o nome de model1.xsd, validado no programa comercial XML Spy, como ilustram as Figuras 5 e 6.

A partir do arquivo GML, utilizou-se a segunda ferramenta construída para geração do arquivo no formato FDE. Como entrada foi passado o arquivo model1.xsd e a saída, mostrada a seguir, foi salva como model1.txt.

```
0001                    *E
0001                  Rio
0001                Ponte
0001                 RHid
0001           Rio: linha
0001             Rio: Vaz
0001         Ponte: ponto
0001          RHid: linha
0001       RHid: poligono
```

6 Conclusões

A utilização de padrões de análise pode contribuir para a melhoria dos modelos conceituais de BDG, uma vez que são soluções já testadas e aprovadas anteriormente, diminuindo o tempo necessário para o projeto conceitual e também a possibilidade de inserção de erros. Outrossim, se estes padrões de análise já estiverem codificados em uma linguagem própria para intercâmbio e armazenamento de informações geográficas, a utilização dos mesmos torna-se ainda mais viável.

Para alcançar os benefícios acima apontados, primeiramente, este trabalho especificou um conjunto de regras para a codificação de um modelo conceitual de BDG baseado no *framework* conceitual GeoFrame para a linguagem de marcação GML. Nesta etapa do trabalho, destacam-se as regras desenvolvidas para esta tradução, especificamente aquelas relacionadas ao mapeamento dos construtores tipo tema (pacote), conceito antes inexistente na GML. Uma vez definidas as regras de conversão, um protótipo foi construído implementando estas regras, de modo a automatizar a geração de código GML a partir de um modelo conceitual UML-GeoFrame.

Uma vez codificados em GML, os construtores UML-GeoFrame podem ser (re)utilizados em outras modelagens, bem como transcritos para outros modelos de dados, sem a necessidade de alteração do modelo conceitual. Isto contribui para a interoperabilidade entre diferentes sistemas de informação geográfica, tanto em nível de dados quanto em nível de projeto, que foi o foco deste trabalho.

Tendo os modelos codificados em um formato especialmente criado para intercâmbio, resta ainda a tarefa de minerar os esquemas a fim de obter os candidatos a padrão de análise. Deste modo, utilizando-se um formato de entrada de dados existente para ferramentas de MD, restrito, mas já testado, foi necessária a especificação de regras de tradução da GML para este formato. Uma vez determinadas as regras, foi construída uma ferramenta que gera, automaticamente, o arquivo pronto para mineração, partindo-se de um arquivo GML.

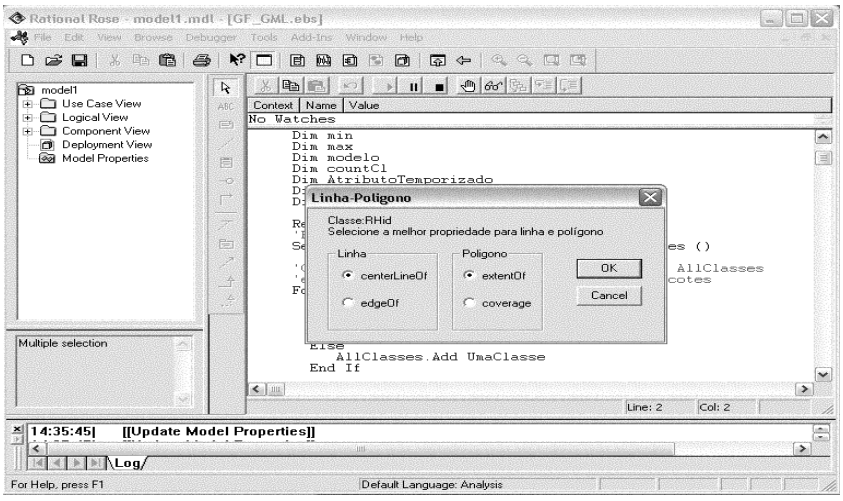

Figura 5 - Script gerador de GML rodando no Rational Rose

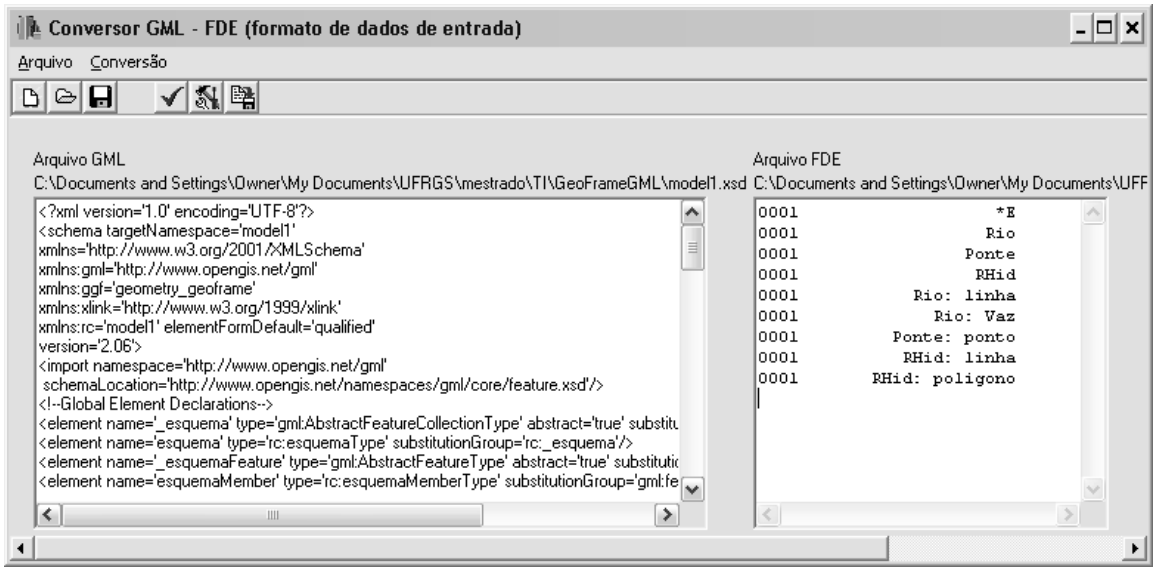

Figura 6 - Ferramenta conversora GML - FDE

Para dar continuidade ao trabalho apresentado, neste artigo, alguns trabalhos futuros fazem-se necessários. Em primeiro lugar, construir uma ferramenta de software capaz de mapear para GML não somente modelos conceituais desenvolvidos com base no *framework* GeoFrame, mas também nos demais formatos citados na introdução desta apresentação. Para isso, deve-se seguir o conceito de conjunto união de construtores, apresentado em [1]. Um segundo trabalho futuro, de importância para a correta preparação dos dados para mineração deve tratar da interoperabilidade semântica entre os modelos. Neste contexto, deve-se investigar uma ontologia para modelagem geográfica.

Referências

[1] Bassalo, G.H.M., Iochpe, C., Bigolin, N. "Representando esquemas no Formato Atributo-Valor para a Inferência de Padrões de Análise". *In: IV Brazilian Symposium on GeoInformatics - GeoInfo 2002*. Caxambu, Brazil, 2002.

[2] Borges, K.V.A., Davis, C.A., Laender, A.H.F. "OMT-G: An Object Oriented Data Model for Geographic Applications". GeoInformatica 5(3): 221-260, 2001.

[3] Carlson, D. "Modelagem de aplicações XML com UML". São Paulo, Makron Books, 2002.

[4] Fayyad, U. M.; Piatetsky-Shapiro, G.; Smyth, P. "From Data Mining to Knowledge Discovery in Databases". AI Magazine, v.17, n.3, p.37-54, fall 1996.

[5] Gamma, E; et al. "Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos". Porto Alegre: Bookman, 2000.

[6] Köesters, B., Pagel, B., Six, H., "GIS-application development with GeooOA". *International Jouranl of Geographic Information Science*. Londres, Inglaterra, 1997.

[7] Lisboa F., J., Iochpe, C., "Specifying analysis patterns for geographic databases on the basis of a conceptual framework". In *Proc.7th ACM GIS*, Kansas City, 1999.

[8] Lisboa F., J., "Desenvolvimento de uma ferramenta CASE para o Modelo UML-GeoFrame com suporte para padrões de análise". *In: IV Brazilian Symposium on GeoInformatics - GeoInfo 2002*. Caxambu, Brazil, 2002.

[9] OpenGIS Consortium. "Geography markup Language (GML) 2.0. Open GIS Implementation Specification, 2001". http://www.opengis.net/gml/01-029/GML2.html.

[10] Parent, C., Spaccapietra, S., Zimanyi, E., "MADS: Modeling of Application Data with Spatio-Temporal features". http://lbdwww.epfl.ch/e/research/mads.

[11] Qin, J.; Paling, S. "Converting a controlled vocabulary into an ontology: the case of GEM", *Information Research 6,* 2001.

[12] Rocha, L. V. ; Edelweiss, N.; Iochpe, C. "GeoFrame-T: A Temporal Conceptual Framework for Data Modeling". In: ACM Symposium on Advances in GIS, 2001, Atlanta. Proc.

[13] Rational Rose 2000e. "Rose Extensibility User's Guide", 2000.

[14] Ministry of sustainable resource management. "SAIF 3.2. Geographic Data BC", Province of British Columbia, Canadá, 2001. Disponível em http://home.gdbc.gov.bc.ca/SAIF.

[15] Silva, C.M.S, Iochpe, C., Engel, P.M., "Using Knowledge Discovery in Database to Identify Analysis Patterns", *5th International Conference on Enterprise Information System*, Angers, France, 2003.