



**MINISTÉRIO DA DEFESA  
EXÉRCITO BRASILEIRO  
DCT - DSG  
1º CENTRO DE GEOINFORMAÇÃO  
(Comissão da Carta Geral do Brasil / 1903)**

**RELATÓRIO TÉCNICO N° 04/2019 – DGEO/1ºCGEO**

**VISTO**

**Chefe do 1ºCGEO**

**PADRÃO DE MAPEAMENTO ENTRE MODELAGENS**

**1. FINALIDADE**

1.1. Este documento tem por finalidade de apresentar uma proposta de arquivo de mapeamento padrão entre modelagens de dados geoespaciais, permitindo de forma flexível a conversão entre especificações e formatos.

**2. REFERÊNCIAS**

2.1. Diniz, Felipe de Carvalho, *Composition of Semantically Enabled Geospatial Web Services*, Dissertação de Mestrado, 2016.

2.2. Relatório Técnico nº 02/2018 – DGEO/1ºCGEO – Atividades de Desenvolvimento Executadas na Reambulação de Santa Maria/RS.

**3. INTRODUÇÃO**

3.1. Tradicionalmente, as OMDS/DSG executam a conversão de modelagens e formatos de dados geoespaciais por intermédio do *software* FME Workbench, sendo utilizado, por exemplo, no mapeamento de EDGV 2.1.3 MDB para EDGV 2.1.3 Gothic e de EDGV 2.1.3 Gothic para EDGV 2.1.3 SHP. Tal forma de mapeamento é extensa e propensa a erros, e em 2015 foi utilizado um arquivo auxiliar no formato *CSV* para definir as regras de mapeamento, utilizando o *transformer PythonCaller* no FME, realizando o mapeamento entre EDGV 2.1.3 DSGTools e EDGV 2.1.3 Gothic, e EDGV 2.1.3 MDB para EDGV 2.1.3 SQLite. A figura 1 apresenta o mapeamento por intermédio de múltiplos *transformers* no FME, e um arquivo de mapeamento no formato *CSV*.



4.2. O arquivo de mapeamento é dividido em duas partes, mapeamento de classes e mapeamento de atributos. O mapeamento de atributos permite mudar o nome de atributos independentemente da classe que ele pertence, e também permite a tradução de valores dos atributos, ou seja, a mudança de certos valores.

4.3. O mapeamento usa o conceito de atributos de ida e de volta (*attr\_ida* e *attr\_volta*), sendo a direção de mapeamento padrão de *ida* para *volta*. Esse sentido padrão é chamado de *mapeamento de ida*, e o sentido contrário é chamado de *mapeamento de volta* (de *volta* para *ida*). Analogamente, também são utilizados valores de ida e de volta (*valor\_ida* e *valor\_volta*). Uma determinada tradução também pode ocorrer em só um dos sentidos, tendo um atributo específico para restringir a tradução. Os exemplos 1, 2 e 3 apresentam mapeamentos de atributos, com e sem tradução, e com traduções que só ocorrem em um determinado sentido.

```
{
  "attr_ida": "geometriaaproximada",
  "attr_volta": "geometriaAproximada"
}
```

Mapeia os valores do atributo *geometriaaproximada* para o atributo *geometriaAproximada*.

Exemplo 1 – Mapeamento simples de atributo.

```
{
  "attr_ida": "jurisdicao",
  "attr_volta": "jurisdicao",
  "traducao": [
    {
      "valor_ida": 6,
      "valor_volta": 8
    }
  ]
}
```

Mapeia os valores do atributo *jurisdicao*, mudando o valor 6 para 8, enquanto mantendo os valores restantes iguais.

Exemplo 2 – Tradução de atributo.

```
{
  "attr_ida": "regime",
  "attr_volta": "regime",
  "traducao": [
    {
      "valor_ida": 1,
      "valor_volta": 2,
      "sentido": "volta"
    },
    {
      "valor_ida": 3,
      "valor_volta": 4,

```

```

    "sentido": "volta"
  }
]
}

```

Mapeia os valores do atributo *regime*, mudando os valores 2 para 1 e 4 para 3 quando o mapeamento for de volta. No mapeamento de ida os valores não são alterados. Os valores restantes não são alterados.

#### Exemplo 3 – Tradução de atributo com direção.

4.4. No mapeamento entre classes são definidas classes de ida e de volta (*classe\_ida* e *classe\_volta*). Além da tradução de classes podem ser definidos valores padrão para os atributos (*atributos\_default\_ida* e *atributos\_default\_volta*) e também podem ser feitas traduções específicas para a classe seguindo o mesmo padrão do que foi apresentado no item 4.3. A geometria das classes não é apresentada (removendo assim sufixos e prefixos de geometria). Os exemplos 4, 5, 6 e 7 apresentam mapeamentos de classes, com e sem valores *default*, e com traduções de atributos.

```

{
  "classe_ida": "cb.hid_foz_maritima",
  "classe_volta": "Foz_Maritima"
}

```

Mapeia as feições da classe *cb.hid\_foz\_maritima* para a classe *Foz\_Maritima*. Isso é feito para todas as primitivas geométricas de *cb.hid\_foz\_maritima*.

#### Exemplo 4 – Mapeamento simples de classe.

```

{
  "classe_ida": "edgv.cobter_area_edificada",
  "classe_volta": "Area_Edificada",
  "atributos_default_ida": [
    {
      "nome_atributo": "geometriaAproximada",
      "valor": 1
    }
  ]
}

```

Mapeia as feições da classe *edgv.cobter\_area\_edificada* para a classe *Area\_Edificada* definindo o valor 1 para o atributo *geometriaAproximada* quando o sentido do mapeamento for de ida.

#### Exemplo 5 – Mapeamento de classe com valores *default*

```

{
  "classe_ida": "cb.hid_ilha",
  "classe_volta": "Ilha",
  "traducao_atributos": [
    {
      "attr_ida": "tipoilha",
      "attr_volta": "tipoElemNat"
    }
  ]
}

```

Mapeia as feições da classe *cb\_hid\_ilha* para a classe *Ilha*, mapeando os valores do atributo *tipoilha* para o atributo *tipoElemNat*.

#### Exemplo 6 – Mapeamento de classe com tradução de atributos.

```
{
  "classe_ida": "cb.hid_trecho_drenagem",
  "classe_volta": "Trecho_Drenagem",
  "traducao_atributos": [
    {
      "attr_ida": "coincidecomdentrode",
      "attr_volta": "coincideComDentroDe",
      "traducao": [
        {
          "valor_ida": 15,
          "valor_volta": 1,
          "sentido": "ida"
        },
        {
          "valor_ida": 97,
          "valor_volta": 24,
          "sentido": "volta"
        }
      ]
    }
  ]
}
```

Mapeia as feições da classe *cb\_hid\_trecho\_drenagem* para a classe *Trecho\_Drenagem*, mapeando os valores do atributo *coincidecomdentrode* para o atributo *coincideComDentroDe*, mudando os valores de 15 para 1 somente no sentido de ida, e 24 para 97 somente no sentido de volta. Os valores restantes não são alterados.

#### Exemplo 7 – Mapeamento de classe com tradução de atributos e valores.

4.5. O próximo conceito a ser apresentado é o de filtros de ida e de volta (*filtro\_ida* e *filtro\_volta*). A utilização do filtro é necessária quando somente um subconjunto de uma classe deve ser mapeado. Somente feições que atendam todas as condições definidas nos filtros serão mapeadas (respeitando o sentido de mapeamento). Os filtros são definidos por pares atributo/valor e operadores *AND*, *OR* e *NOT*. Os exemplos 8 e 9 apresentam mapeamentos de classes com filtros de atributos simples e complexos.

```
{
  "classe_ida": "cb.loc_cidade",
  "classe_volta": "Cidade",
  "filtro_volta": {
    "nome_atributo": "tipoCidade",
    "valor": 1
  },
  "atributos_default_ida": [
    {
```

```

        "nome_atributo": "tipoCidade",
        "valor": 1
    }
]
}

```

No sentido de volta, mapeia as feições da classe *Cidade*, com filtro de feições onde *tipoCidade* deve ser 1, para a classe *cb.loc\_cidade*. No sentido de ida mapeia feições da classe *cb.loc\_cidade* para a classe *Cidade*, definindo o valor 1 para o atributo *tipoCidade*.

Exemplo 8 – Mapeamento de classe com filtro de volta.

```

{
  "classe_ida": "cb.loc_capital",
  "classe_volta": "Cidade",
  "filtro_volta": {
    "$or": [
      {
        "nome_atributo": "tipoCidade",
        "valor": 2
      },
      {
        "nome_atributo": "tipoCidade",
        "valor": 3
      }
    ]
  },
  "traducao_atributos": [
    {
      "attr_ida": "tipocapital",
      "attr_volta": "tipoCidade"
    }
  ]
}

```

No sentido de volta, mapeia as feições da classe *Cidade*, com filtro de feições onde *tipoCidade* deve ser 2 ou 3, para a classe *cb.loc\_capital*. No sentido de ida mapeia feições da classe *cb.loc\_capital* para a classe *Cidade*. Além disso é feita a tradução do atributo *tipocapital* para *tipoCidade*.

Exemplo 9 – Mapeamento de classe com filtro de volta complexo.

4.5.1. Para separar uma classe pelo tipo de geometria deve ser utilizado um filtro com o atributo reservado *\$GEOM\_TYPE*.

4.6. O último conceito a ser apresentado é o mapeamento de tuplas de atributos, que permite um mapeamento N:M entre valores de atributos. Tal opção é definida pela chave *mapeamento\_atributos* interna ao mapeamento de classes, e permite definir tuplas de atributos de ida e de volta (*tupla\_ida* e *tupla\_volta*), onde cada tupla é um conjunto de um ou mais pares de atributo/valor. No mapeamento de ida, se a feição atender todos os pares de atributo/valor do conjunto de tuplas de ida o mapeamento é executado, recebendo todos os pares de atributo/valor da *tupla\_volta*. O mapeamento é análogo no caso de volta. O exemplo 10 apresenta um caso de

mapeamento de tuplas de atributos.

```
{
  "classe_ida": "edgv.cobter_vegetacao",
  "classe_volta": "Veg_Cultivada",
  "filtro_ida":
  {
    "nome_atributo": "tipo",
    "valor": 102
  },
  "mapeamento_atributos": [
    {
      "sentido": "ida",
      "tupla_ida": [
        {
          "nome_atributo": "tipo",
          "valor": 102
        }
      ],
      "tupla_volta": [
        {
          "nome_atributo": "tipoLavoura",
          "valor": 1
        },
        {
          "nome_atributo": "classificacaoPorte",
          "valor": 1
        }
      ]
    }
  ]
}
```

O exemplo é uma simplificação do caso real, só devendo ser considerado no sentido de ida, onde são mapeadas as feições da classe *edgv.cobter\_vegetacao*, com filtro de feições onde *tipo* deve ser *102*, para a classe *Veg\_Cultivada*, realizando o mapeamento de atributos quando *tipo* tem valor *102* para os valores *1* para *tipoLavoura* e *1* para *classificacaoPorte*.

#### Exemplo 10 – Mapeamento de atributos (tuplas).

4.7. O arquivo de mapeamento utiliza um conceito de cascata de mapeamento, executando as conversões na seguinte ordem: mapeamento de atributos, definição de atributos *default*, tradução de atributos e, finalmente, mapeamento de atributos (tuplas). Desta forma, o mapeamento de atributos pode ser executado de forma geral inicialmente para todas as classes, e uma classe específica pode ter um nome de atributo ou valor distinto utilizando as outras formas de mapeamento.

4.8. De modo a formalizar a definição do arquivo de mapeamento, a Sequência 1 apresenta a forma aumentada de Backus-Naur (*Augmented BNF*) da notação do mapeamento.

Sequência 1 – Definição em *Augmented BNF* do padrão de mapeamento.

```

<mapeamento>::= {
    "mapeamento_classes": [1*n<mapeamento de classes>],
    "mapeamento_atributos": [0*n<mapeamento de atributos>]
}

<mapeamento de atributos>::= {
    "attr_ida": "string",
    "attr_volta": "string"
    0*1<grupo tradução>
}

<grupo tradução>::= , "traducao": [1*n<tradução>]

<tradução>::= "valor_ida": <valor>,
    "valor_volta": <valor>
    0*1<define sentido>

<define sentido>::= , "sentido": "<sentido>"

<mapeamento de classes>::= {
    "classe_ida": "string",
    "classe_volta": "string"
    0*1<atributo default ida>
    0*1<atributo default volta>
    0*1<filtro ida>
    0*1<filtro volta>
    0*1<tradução de atributos>
    0*1<mapeamento att classe>
}

<atributo default ida>::= , "atributo_default_ida": [1*n<atributo valor>]
<atributo default volta>::= , "atributo_default_volta": [1*n<atributo valor>]

<atributo valor>::= {
    "nome_atributo": "string",
    "valor": <valor>
}

<tradução de atributos>::= , "traducao_atributos": [1*n<mapeamento de atributos>]

<filtro ida>::= , "filtro_ida": <filtro>
<filtro volta>::= , "filtro_volta": <filtro>

<filtro>::= <operador>|<atributo valor>

<operador>::= <operador or>|<operador and>|<operador not>

<operador or>::= {
    "$or": [1*n<filtro>]
}

<operador and>::= {
    "$and": [1*n<filtro>]
}

<operador not>::= {
    "$not": <filtro>
}

<mapeamento att classe>::= , "mapeamento_atributos": [1*n<grupo mapeamento>]

<grupo mapeamento>::= "tupla_ida": [1*n<atributo valor>],
    "tupla_volta": [1*n<atributo valor>]
    0*1<define sentido>

<sentido>::= ida|volta

```



<valor> ::= null|inteiro|"string"

4.9. Vale salientar que o mapeamento é executado por feição, onde uma feição na modelagem origem só pode ser mapeada para uma feição na modelagem destino.

4.10. Foi implementado em Python o conversor entre modelagens que utiliza o modelo proposto para realizar o mapeamento. O código está disponível no Github, no seguinte endereço:

[https://github.com/dsgoficial/conversao\\_modelagens](https://github.com/dsgoficial/conversao_modelagens)

## 5. PERSPECTIVAS FUTURAS

5.1. Como perspectiva futura, o 1º CGEO envidará esforços para criar mapeamentos bidirecionais entre outras modelagens, permitindo uma série de conversões entre especificações e formatos. Na figura 2 está apresentada a estrutura de conversão entre especificações, modelagens e formatos planejadas. Em vermelho é a conversão atual do Gothic para SHP implementada no FME para carregamento no BDGEx.

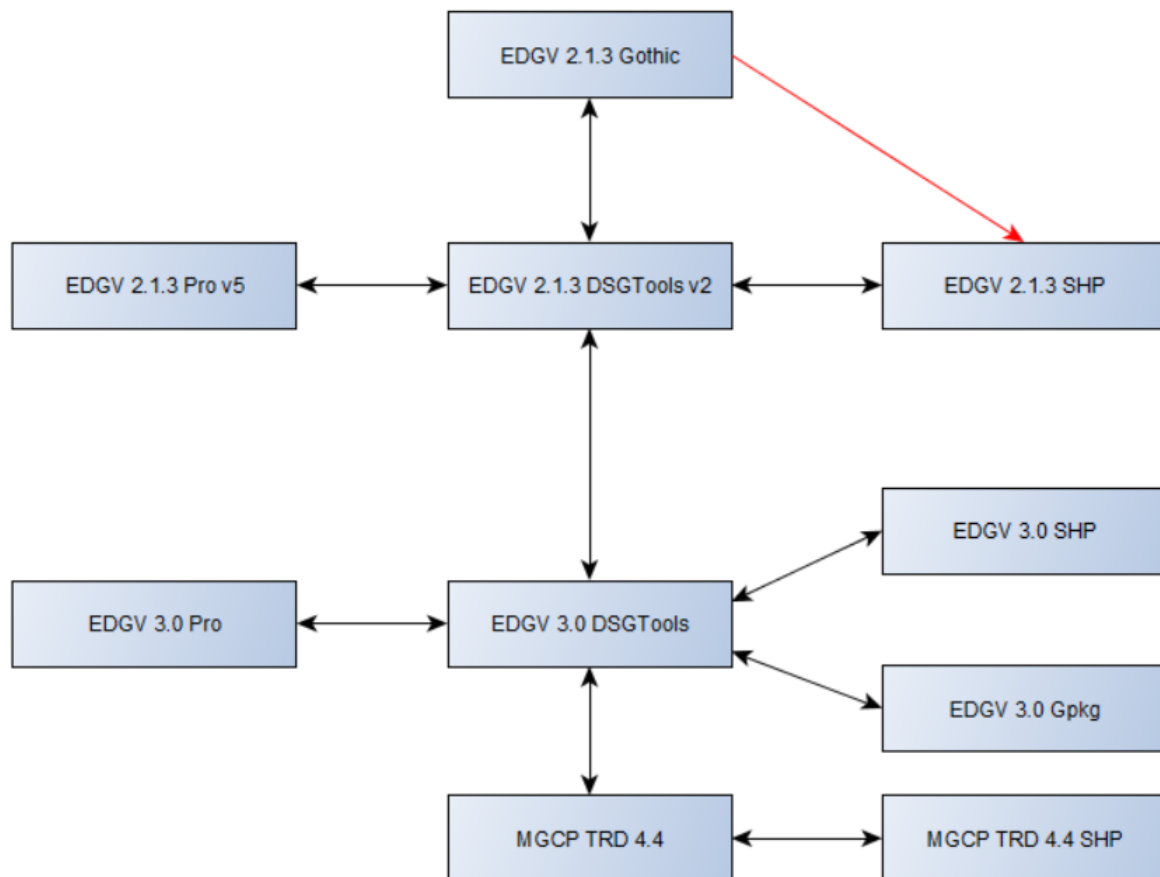


Figura 2 – Conversões previstas.

5.2. A conversão entre EDGV 3.0 DSGTools e MGCP TRD 4.4 foi proposta como tema de Projeto Interdisciplinar no Curso de Cartografia e Sistemas de Informações Geográficas do 2º CGEO. Se

encontra em desenvolvimento no 1º CGEO o mapeamento entre EDGV 2.1.3 Pro v5 e EDGV 2.1.3 DSGTools v2, e entre EDGV 2.1.3 DSGTools v2 e EDGV 2.1.3 SHP.

5.3. O mapeamento EDGV 2.1.3 Gothic e EDGV 2.1.3 DSGTools v2 já está completo e atualmente o 1º CGEO está exportando todos os dados vetoriais validados do Gothic para o PostGIS, que será o banco de produtos legados. O 1º CGEO sugere que, com a futura obsolescência do Gothic/Lamps para validação e edição, todos os dados validados no Gothic sejam armazenados na EDGV 2.1.3 DSGTools v2.

5.4. A equipe do DSGTools está desenvolvendo uma ferramenta para conversão de modelagens e formatos, que utilizará o formato proposto para mapeamento entre modelagens.

5.5. Em teoria, tal conceito de mapeamento também poderia ser integrado ao BDGEx na forma de serviço, permitindo que tal serviço disponibilize dinamicamente o mesmo arquivo em múltiplas modelagens para o usuário, conforme sua necessidade.

## 6. CONCLUSÃO

6.1. O padrão proposto de mapeamento permite a conversão bidirecional entre modelagens, especificações e formatos, tendo expressividade o suficiente para realizar mapeamentos complexos como entre a EDGV 3.0 e o padrão do MGCP TRD 4.4.

6.2. Uma vez todos os mapeamentos bidirecionais implementados, as DGEO terão uma grande flexibilidade na disponibilização e manutenção de formatos e modelagem. Pela forma com que o processo foi implementado, pode-se utilizar o mapeamento por intermédio do FME (utilizando PythonCaller) ou diretamente na ferramenta de conversão do DSGTools. Também permite que mapeamentos sejam executados em série, permitindo que todas as combinações de conversão sejam executadas.

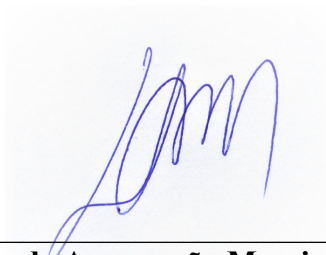
Porto Alegre – RS, 02 de maio de 2019.



**Felipe de Carvalho Diniz – Cap**

Supervisor da Célula de Controle de Qualidade Interno  
1º Centro de Geoinformação

**De Acordo:**



**Leonardo Assumpção Moreira – Maj**  
Chefe da Divisão de Geoinformação  
1º Centro de Geoinformação