Architecture of the Open Format Converter

Dariusz Mrozek*, Jacek Frączek**

* Department of Computer Science, Silesian University of Technology, ul. Akademicka 16, 44-100 Gliwice, Poland, tel. +48 32 2371339, fax +48 32 2372733, e-mail mrozek@zti.iinf.polsl.gliwice.pl
** Department of Computer Science, Silesian University of Technology, ul. Akademicka 16, 44-100 Gliwice, Poland, tel. +48 32 2371339, fax +48 32 2372733, e-mail jacekf@polsl.gliwice.pl

1 Introduction

Geographic information systems (GIS) are specialized systems that allow to visualize and analyze earth phenomena, prepare planning strategies, predict and explain events and processes [1, 3]. The analysis of spatial information in Earth Sciences often involves data that is distributed in many places and stored in various formats. Many proprietary standards (and versions of these standards) have been designed to store spatial information. That is why, the transfer of data between different systems is a very common problem. The issues of exchange of vast amounts of spatial data, often accompanied by non-spatial (descriptive, quantitative) information are often a source of severe difficulties. Another type of troublesome conversions is vector-to-raster and raster-to-vector conversions described *inter alia* in [2].
Many organizations and government agencies use special formats to collect data due to their own areas of interest, e.g.: Digital Line Graphs (DLG) used the by US Geological Survey (USGS), Topologically Integrated Geographic Encoding and Referencing Files (TIGER) used by the US Census Bureau and Spatial Data Transfer System (SDTS) developed by the US government. Many formats have been designed to facilitate the interchange of information between systems [4] – Drawing Interchange Format (DXF) linked to the CAD/CAM applications, MapInfo Data Transfer Files (MIF/MID) – a desktop mapping system used by MapInfo, MicroStation Design Files (DGN) – an internal format used by Bentley Systems Inc.'s MicroStation, and Arc Export used to transfer files between different versions of ARC/INFO. There are also many standardization initiatives undertaken by the international organizations for improving interoperability between GIS systems, like: OpenGIS Geography Markup Language (GML) [6], W3C Scalable Vector Graphics (SVG) [5], etc.
The paper presents the architecture of the Open Format Converter, which enables the conversion of vector geospatial data to various formats. The Open Format Converter defines the framework for multipurpose converter that is easy to extend far beyond the proposed, implemented geospatial formats. The hereby presented architecture was used as the foundation for a converter application that was developed in Department of Computer Science of Silesian University of Technology for the Polish State Committee for Scientific Research (KBN) project: *Implementation of Regional Spatial Information System in Silesia Province* on order of Institute of Spatial and Cadastre Systems (ISPiK S.A.)

2 The Architecture of the Converter

The general architecture of the converter is presented on Fig.1. The main component of the architecture is a Translation Core (**TC**), which consists of two modules: Geo-Core (**GC** – *geospatial core*) and Attribute Matrix (**AM**).

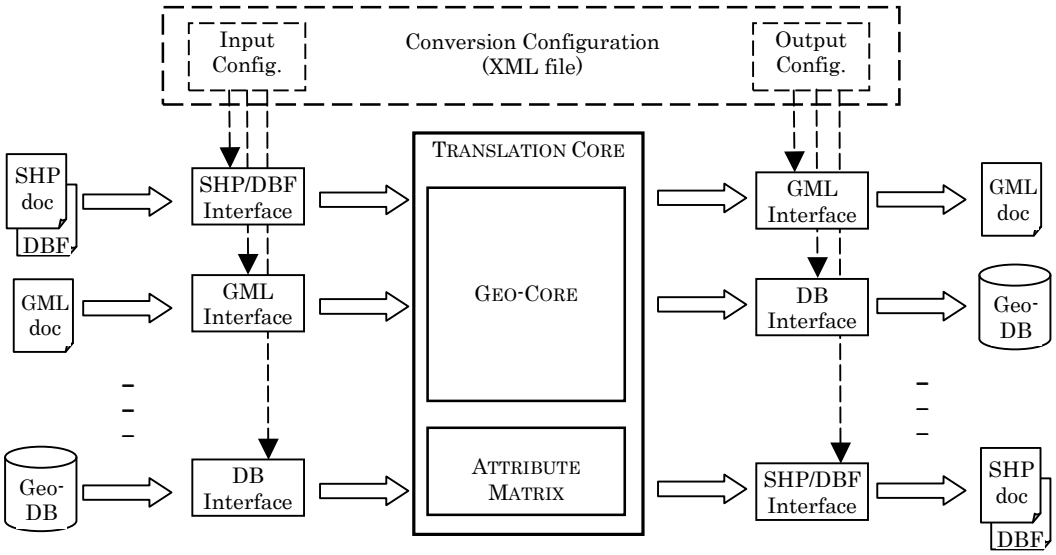

Figure 1: Architecture of the Open Format Converter

Geo-Core is a universal container for vector spatial 2- and 3-dimensional objects. Its structure was designed under the assumption of compatibility with the GML 2.0 specification [7]. Fig.2 depicts the inheritance structure of objects intended to store spatial data: points, lines, rectangles, polygons (with and without holes) and sets of spatial objects: multi-points, multi-lines, multi-polygons, multi-objects, etc. The structure of more compound objects incorporates simpler objects (e.g.: multi-polygons consist of polygons, which are the set of directed linear rings, which are defined as ordered set of points). Fig.3 shows the schema of aggregation dependences between the structures of various objects stored in GC.

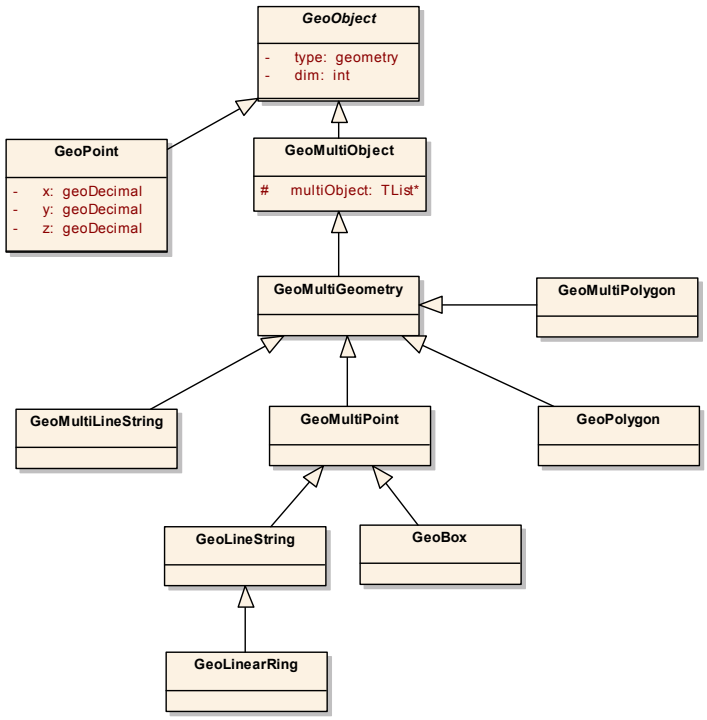

Figure 2: The structure of inheritance of Geo-Core objects (only the most basic members and methods are shown)

Attribute Matrix is an additional container for non-spatial data. This type of data is stored in the form of *attribute-value* pairs. Attributes represent additional information linked

with a spatial object, e.g. the name of a street, or the average depth of a water reservoir. AM is also able to store other types of data like object's properties and features or even display formatting information (colours, line widths, line styles), which specifies how a spatial object should be presented on a map. The data that is stored in AM module may come directly from the input data sets or may be calculated on-the-fly by the configuration process.
During the process of conversion, GC is intended to store one *currently processed* object. The AM module is loaded with a list of attributes related to the current object. To provide flexible means of bulk translation of many spatial objects, a special iterative procedure must be constructed. In the simplest predefined case, it fetches data from the source (input interface) and moves it to the destination (output interface). More complex implementations include the option to divide the output data into smaller packages.

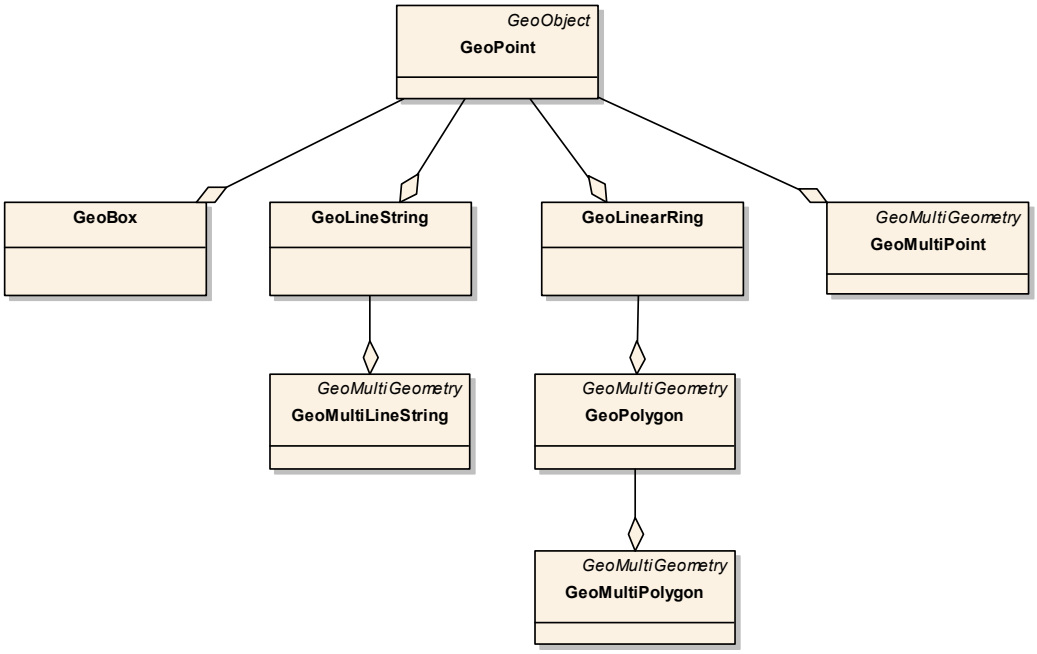

Figure 3: Schema of aggregation of geospatial objects (without the GeoMultiGeometry object which may consist of objects of any type)

The structure of the translation framework is surrounded by the extensible set of interfaces responsible for accessing data stored in various formats. Input and output interfaces are independent modules that co-operate with TC. The design based on the separation of input and output interfaces simplifies the implementation of access modules, because it imposes the requirement to implement only one-direction translation: from the source format to the TC format or from the TC format to the destination format.
The Translation Core therefore defines a framework, to which appropriate input/output interfaces are connected. Input interfaces read data from files or/and from databases and store it inside the TC modules. It is possible to write interfaces that read data from many sources, e.g. when converting from ESRI Shape format, it is possible to read spatial information from SHP file and non-spatial information from DBF files. Configurations of mixed or distributed data sources are also possible. Output interfaces may write data to files and/or databases. Fig.4 shows the basic inheritance diagram for input and output interfaces.
In many cases the data conversion process requires additional configuration. This can be accomplished by incorporating into the application the possibility to use external configuration files. For that purpose the Open Format Converter uses special XML configuration files (the structure of the configuration file will be presented in section 4).

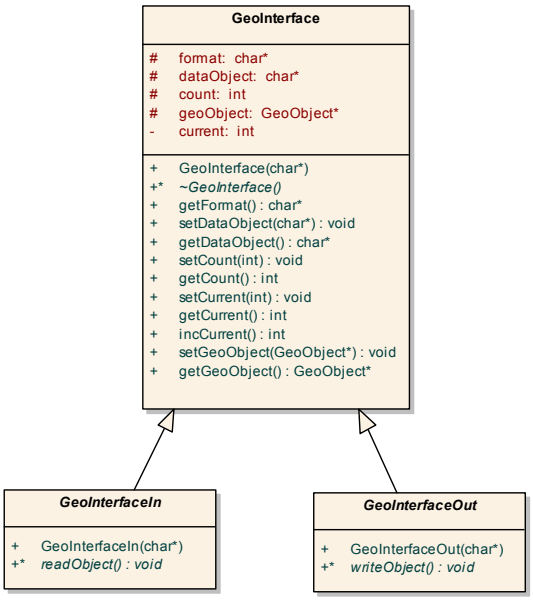

Figure 4: The Schema of inheritance for access interfaces to objects without non-spatial data

The main advantage of the advanced configuration is the possibility to influence and change the behaviour of the basic conversion path. Appropriate configuration can modify activities carried out by input and/or output interfaces. For every single type of conversion (e.g., SHP/DBF → GML) one configuration file should exist.

3 Conversion course and configuration process

Specification of calling parameters is the simplest way of controlling the behaviour of the converter application. It is sufficient only in the most straightforward situations, when no specific transformations are needed (when objects are transferred to the destination without any additional changes). However, non-spatial data often has to be handled in a specific way – it is frequently a subject to various transformations or even special actions executed upon its attributes. A flexible way of controlling the converter activities is the use of external configuration files.

3.1 Simple conversion

The so-called, *simple* conversion process (Fig.5) may be controlled by the use of parameters.

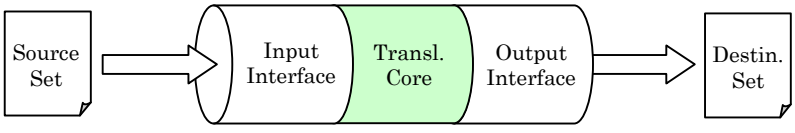

Figure 5: Simple conversion process

Conversion is executed according to a very simple principle – appropriate interfaces are defined, the data is read from the source set and written in the destination format. No transformations are done to the data. In this case, Input Interface is responsible for interpreting and embedding spatial data and non-spatial attributes in the GC/AM structure. Output Interface is responsible just for reading information from GC/AM and writing it in the destination format.

3.2 Advanced configuration of the input interface

The configuration of the converter allows to specify various kinds of transformations that should be carried on the non-spatial data during the conversion process. The transformations include:

- Attributes' type conversion (e.g., boolean *False/True* to numeric *0/1*).
- Trimming, truncating, and rounding data.
- Transforming characters to uppercase/lowercase.
- National language character-set conversions.
- Setting values for attributes from the source data that have missing values or should have assigned specific constant values in the output data.
- Other transformation formulae for source/destination values.

Fig.6 shows the conversion process with special configuration of the input interface. In the example, some additional information has to be acquired from the other data set referenced from the main data set.

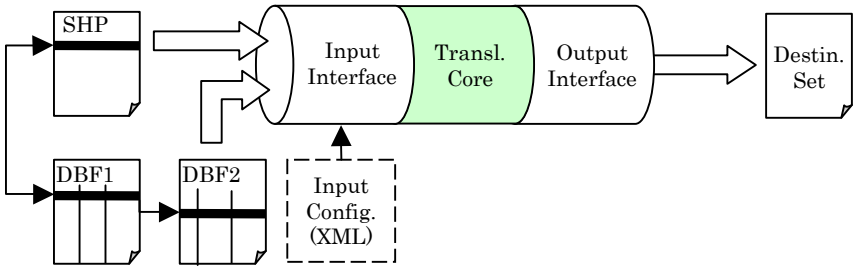

Figure 6: Configuration of the input interface

A shape file (SHP) containing geometry and a corresponding attribute set (DBF1) play the role of the principle source data set. Additional attribute data set DBF2 plays the role of the supplementary source linked to the main source with appropriate join condition. Join operation is performed with the use of an *id* field (e.g., *riverID*) from DBF1 which references a record in the lookup table DBF2 (e.g., identified by *DBF2.riverID*). *DBF1.riverID* is a foreign key to the *DBF2.riverID*. The system allows to retrieve the value of any field in additional data set (e.g., *riverName*, *riverLength*).
This advanced type of conversion requires the operator to prepare a special configuration, which specifies:

- The supplementary source (in the case of files: path and name of the file).
- Fields that are involved in the join condition.
- Name of the additional field, which value has to be retrieved from the supplementary data set.

3.3 Advanced configuration of the output interface

Configuration of the output interface is associated with retrieving data from TC and writing it in the specified destination format. In this case – before data is loaded into the output set – additional changes, transformations, corrections and references to other data sources may take place. Fig.7 shows such an example situation.

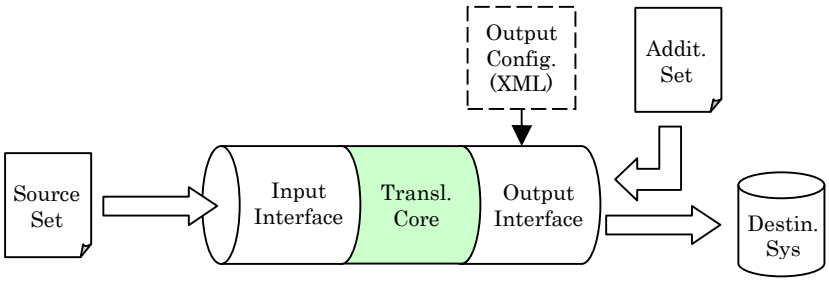

Figure 7: Configuration of the output interface

In the presented example, spatial and non-spatial data stored in the TC module is to be inserted into a destination system. The information stored in a configuration file controls the way the output interface runs. In the case, when the destination system is a database, additional configuration specifies:

- Information required to connect to the database (e.g., connect string).
- Names of tables, to which data should be inserted.
- Other features of the destination system (e.g., how often *commit* operation should be submitted, job scheduling, etc.).

The *Additional Set* on Fig. 7 holds additional display information (e.g. colour, line style, etc.) that should be incorporated into the output data. In this case, content of the configuration file must contain information specific for the process of reading additional referenced attributes (it is analogous to the case of the advanced configuration of the input interface).

4 Structure of the configuration file

The flexible usage of input and output interfaces is based on the application of advanced configuration options. The use of configuration files helps avoid recompilation of the application modules by storing the changing parameters of the conversion process outside the interfaces' code.

The Open Format Converter uses configuration files in XML format. The example contents of the file for the *shape* to *GML* conversion process is presented below:

```
<?xml version="1.0" encoding="UTF-8"?>
<Configuration>
   <General>
      <SourceFormat>Shape</SourceFormat>
      <DestinationFormat>OGC-GML</DestinationFormat>
      <TransferGeometry>true</TransferGeometry>
      <TransferAttributes>true</TransferAttributes>
      <GeoType>POINT</GeoType>
   </General>
   <Input>
      <Dictionaries>
         <Dictionary name="CITIES" type="DBF" file=".\rivers.dbf"/>
      </Dictionaries>
      <Attribute dictionary="CITIES">
         <SrsAttName>Name</SrsAttName>
         <DstAttName>City</DstAttName>
         <JoinCondition>MAIN.cityID = CITIES.ID</JoinCondition>
      </Attribute>
   </Input>
   <Output>
      <PartOfDoc parent="RefineryLayer">HydroMap.xml</PartOfDoc>
      <ExplicitCoords>true</ExplicitCoords>
      <SRSName>UTM Zone 10</SRSName>
      <RecNumber>10000</RecNumber>
      <Element name="WaterRefinery">
         <Attribute>
            <SrcAttName>Bio</SrsAttName>
```

```
            <DstAttName>Biological</DstAttName>
         </Attribute>
         <Attribute>
            <SrsAttName>Chem</SrsAttName>
            <DstAttName>Chemical</DstAttName>
            <Value>True</Value>
         </Attribute>
      </Element>
   </Output>
</Configuration>
```

There are three main sections inside the `Configuration` root element: `General`, `Input` and `Output` – each one responsible for specific part of the conversion process. The `General` section is obligatory. It contains common information about the process, like: source format (`SourceFormat`), destination format (`DesinationFormat`) and optional geometry type (`GeoType`). In the presented example, there are also two optional flags: `TransferGeometry` and `TransferAttributes`, which specify whether the geometry and attribute data should be transferred from source to destination set.
The `Input` section is optional and controls the input interface. In the presented example, one supplementary dictionary is declared in order to get the values of additional attributes that are not available in the main attribute set. The value of the `Name` field from the `CITIES` dictionary (specified in the `SrcAttName` tag) is retrieved and put inside the AM module as the value of the `City` attribute (specified in the `DstAttName` tag). Join condition between the main attribute set and secondary `CITIES` attribute set (`MAIN.cityID = CITIES.ID`) is defined in order to point to the appropriate record in the secondary set (`MAIN` is a predefined dictionary and denotes the main attribute set). In this situation, two operations are performed on the additional attribute `Name`. First, the attribute is retrieved from the secondary set, and then its name is changed.
The `Output` section is optional and controls the output interface. The example configuration file contains instructions how geometry coordinates should be presented in the destination set. The `ExpicitCoords` flag decides whether the coordinates are stored in shorter <*gml:coordinates*> or longer format <*gml:coord*>. The `Element` tag describes the name of the generated element in the destination XML document and attributes associated with the element. The `PartOfDoc` tag denotes that generated GML document will be a part of a bigger GML document (`HydroMap.xml`). The `parent` attribute of the `PartOfDoc` tag indicates the tag in the output XML document (`RefineryLayer`), where the data should be placed. The `RecNumber` tag manages the division of the output document into smaller parts and specifies the number of records in the single part of the output file. `SRSName` contains name of the Spatial Reference System. The `Attribute` tags inside the `Element` tag describe transformations that are executed on attributes during the transfer of data from AM module into the specified GML element of the destination document. Attributes that have no dictionary specified, by default denote attributes from `MAIN` dictionary.

5 Summary

The paper presents the architecture of the Open Format Converter, which defines an extensible framework for applications used to transform vector spatial and non-spatial data into required output formats. The architecture uses object-oriented technology to store data in universal internal container and the set of input/output interfaces used to access and write data in various standards. The configuration capabilities allow to control the behaviour of the process and to define additional transformations for non-spatial attributes.

References

[1] Rigaux Ph., Schol M., Voisard A.: Spatial Databases. With Application to GIS. Academic Press, 2002.

[2] Healey R., Dowers S., Gittings B., Mineter M.: Parallel Processing Algorithms for GIS. Taylor & Francis, 1998.

[3] Berry J.K.: Spatial Reasoning For Effective GIS. GIS World, Inc., 1997.

[4] The GeoCommunity, http://www.geocomm.com/.

[5] Scalable Vector Graphics (SVG), Version 1.2, http://www.w3.org/TR/2004/WD-SVG12-20040510.

[6] OpenGIS, Geography Markup Language (GML) Implementation Specification, Version 3.0, http://www.opengis.org/specs.

[7] Geography Markup Language (GML) 2.0. OGC Recommendation Paper, 2001.