Revista Romana de Interactiune Om-Calculator 9(1) 2016, 71-84 © MatrixRom

Visual tools for Software Development in GIS applications

Mihai Gheorghe, Marian Dardala

Bucharest University of Economic Studies
6 Piata Romana, 1st district, Bucharest, 010374 Romania
E-mail: mihai.gheorghe@gdm.ro, dardala@ase.ro

Abstract. This paper aims to showcase a set of features which enables users to develop custom processing models using a specific interface for the workflow. The component presented in this paper, part of the ArcGIS software suite, is called *ModelBuilder* and allows a visual description of the processing flow, which enable users who do not have extensive programming knowledge to define custom processing models. The tool is also addressed for experienced programmers by having the ability to include scripts within the models and setting them parameters for integration with the workflow. This way, the models can be run either as stand-alone tools, on demand using specific interaction routines - GIS Add-ins, or can be embedded in *WebGIS* applications by exporting them as geoprocessing services and run through *ArcGIS Server* or through *Cloud* services provided by ESRI (the company that develops ArcGIS).

Keywords: GIS, human-computer interaction, modeling, ArcGIS ModelBuilder, software engineering, visual development tools.

1. Introduction

The way software applications are being developed has seen a continuous concern growth along with the increasing variety and large scale use of the software. Besides programming specialists, users of software products are also interested in developing techniques to build modules or applications for specific processing needs. Therefore, within software products, components have been designed to enable building functional modules or applications using visual interfaces as an alternative to the classic ones, specific to programmers (the text editors which the programmers write source code in) [2].

It is widely known that programming environments do feature visual tools to define interfaces as well but their role is mainly to increase the productivity of programmers and to enforce a much more precise perspective over the way the application interface is being planned.

Visual development tools are usually found along with software products designed for a wider class of users, including the ones who do not have extensive experience in programming but use software as part of their activity; such as multimedia creative suites which help users to create presentations, GIS (Geographically Information System) based applications which allow creating and editing of maps as well as processing and visualization of data, products for planning and maintaining object-oriented software systems using UML (Universal Modeling Language), tools for designing and maintaining databases.

2. Visual tools for software development

Visual tools for software development have emerged around software products intended for users that do not possess extensive knowledge in programming. An example is the field of creative multimedia software products which enables users to the artistic sense to process images, to generate animations, to edit audio and video sequences in order to include them in various presentations.

In order to build such applications, the developer had to control the presentation's flow, usually by writing code, making this inconvenient for non-specialists. Therefore, *Multimedia Authoring* stands for a particular class of software products designed to build multimedia applications based on two essential concepts: *a metaphor* that underlies the structure of the multimedia project and *pre-programmed software components* which can be directly used in applications and provide a standard functionality linked to the deployed metaphor.

The Metaphor describes the manner in which the components of the presentation are structured. From this perspective, several principles and concepts can be depicted, as following [8].

The Slideshow is a principle which refers to organising the elements in pages, similar to a book, and through interactive routines, the user can browse the content as desired.

The Timeline principle relies on sequencing the presentation elements on a time axis and on providing interactive features, based on a time variable, that allow the user to change the viewing frame and, therefore, the current moment of the application.

A different method to control the deployment of an application is by defining *flowcharts*, also called *workflow diagrams* or *process diagrams*, which enables describing the scenario according to the interaction with the interface components for the presentation.

The pre-programmed elements for this class of software products are either intended to be used as stand-alone (video or audio players), navigation buttons or behaviors are built through scripts and they are attached to interaction elements using drag-and-drop routines.

The use of visual tools for software development is also present in fields which were considered intended solely for experienced software engineers.

The Unified Modeling Language (UML) is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system. It is planned for use with all development methods, life cycle stages and application domains. UML has been considered as a standard in software engineering while it features a framework to integrate a wide variety of diagrams to comprehensively define a software system, allowing virtually all stakeholders, to understand, design, configure, maintain, and control information about such systems [3], [5].

Some diagrams are essential in that they capture the complete scope and behavior of the system and support model translation to code; and some derived: diagrams that show additional views of the essential models [3].

Since the first version of UML was first released, in 1997, a broad range of CASE (Computer-aided software engineering) instruments based on the UML standard emerged and evolved to accomplish an extensive set of software system planning and deployment related tasks; among them, automatically generate source code for virtually all major programming languages. Some tools enable users to construct *reverse engineered models from existing programs*, making non-programming specialists able to take part of the software development process. Also, seamless integrations with acknowledged IDE (Integrated Development Environments) have been implemented such as NetBeans, Eclipse, Visual Studio.

However, *UML diagrams don’t enforce an exhaustive definition of the software system*, therefore, 100% code generation cannot be achieved and they cannot be executed as stand-alone [7].

Nevertheless, *"executable diagrams"* do exist in various software development fields. An example is database creation and management using MySQL Workbench®.

MySQL Workbench is a typical CASE tool for working with MySQL Servers and databases and covers areas such as SQL Development, Data Modeling (Design), Server Administration and Data Migration [11].

It enables users to create and manage connections to database servers and provides the capability to execute SQL queries on the database connections using a built-in SQL Editor.

Also, MySQL Workbench is designed to create models of the database schema graphically using Entity-Relationship Diagrams, *reverse and forward engineer between a schema and a live database*, and edit all aspects of the database using a visual Table Editor. The Table Editor provides easy-to-use facilities for editing Tables, Columns, Indexes, Triggers, Partitioning, Options, Inserts and Privileges, Routines and Views.

In terms of Server Administration, it allows handling of MySQL server instances by managing users, performing backup and recovery, inspecting audit data, viewing database health, and monitoring the performance of the MySQL server.

Migration from acknowledged RDBMS such as Microsoft SQL Server, Microsoft Access, Sybase ASE, SQLite, SQL Anywhere, PostreSQL to MySQL is also supported by MySQL Workbench through GUI routines.

MySQL Workbench is built for the use of both SQL specialists, whose productivity can be improved compared to building and managing a database by writing code, and non-specialists who can easily import or create a MySQL database benefiting from the usability of the interface and automated completeness and consistency checks.

The following section is a presentation of a tool for developing software modules in ArcGIS which mixes the visual interface for describing the processing flow with embedding scripts in the process.

3. Visual tools for operating with GIS

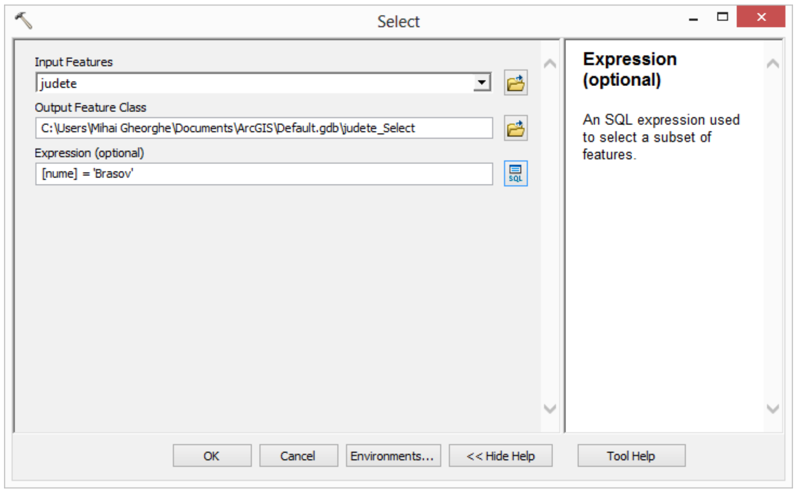
Even since Sofware Engineering was still niche activity, GIS Software has been continuously developed and maintained by a variety of commercial and non-profit organizations [6]. However, ESRI's ArcGIS suite holds more than 40% of the GIS software worldwide markeshare [9]. Therefore, the current study showcases visual tools for operating with GIS provided by the mentioned software.

The ESRI ArcGIS suite is a software product mainly used to edit spatial data, perform spatial analysis and manage GIS data by running geoprocessing tools that are part of the suite.

ArcGIS features a considerable library of geoprocessing tools, called *ArcToolbox*, for accomplishing an extensive number of GIS-specific operations such as spatial analysis, cartography, geocoding but also non-specific tasks such as data management tools.

The instruments from *ArcToolbox* have a standardized Graphic User Interface (GUI) as shown in *Figure 1*, which in most cases enable the user to manually specify input and output data and other variables and parameters relevant to the use of the tool.

While being easy to use and having the possibility to opt for a broad range of geoprocessing instruments, this method of operating with GIS data has the downside of being limited to manually use one tool at a time which in applications that require successive operations for multiple datasets, can be time-consuming and prone to errors.



Input Features: judete
Output Feature Class: C:\Users\Mihai Gheorghe\Documents\ArcGIS\Default.gdb\judete_Select
Expression (optional): [nume] = 'Brasov'

Expression (optional)

An SQL expression used to select a subset of features.

Figure 1. Select by Attribute from ArcToolbox

4. Building Geoprocessing Models

ModelBuilder is a component of ArcGIS designed to create and manage geoprocessing models which automate the use of geoprocessing tools. Models are *“executable flowcharts”* for powerful multi-step GIS processing [10]. Therefore, *ModelBuilder* can also be thought of as a visual programming tool for building workflows.

Geoprocessing models rely on connecting map layers, datasets and other data types with tools and Python scripts into processes. A general format for a process in ModelBuilder is described in *Figure 2.*

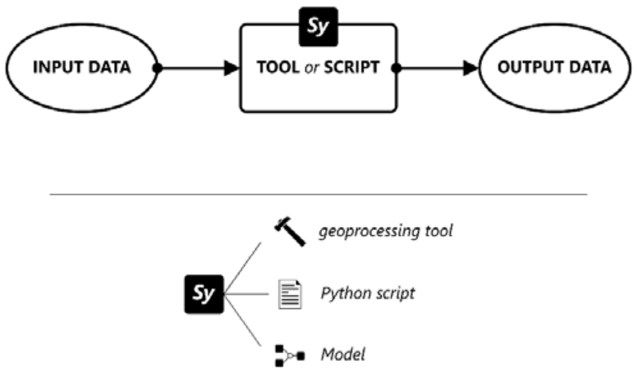


Figure 2. A basic process in ModelBuilder

Models are represented as diagrams that concatenate sequences of processes and geoprocessing instruments, using the output of one process as the input to another process.

Initially, when tools are added to the diagram, the elements are outlined in black and white meaning the model is not yet valid, therefore, it cannot be run in the current state. To overcome this, the tools require having the proper parameters set. After this action is performed, a visual code can easily be noticed:

- Data elements are being represented as ovals while the tools and scripts are illustrated as rectangles
- Input data is colored in blue, Output data is colored in green and the tool is in orange

- The diagram elements are linked with one-way arrows indicating a workflow
- After a successful run of the model, the diagram's elements will have a shadow outline added.

In *Figure 3* a basic model that has the same output as the manual operation illustrated in *Figure 1* has been built and run.

Once saved, a model can be executed anytime needed by double-clicking on it in the *Catalog* window or it can be included in other models with a drag-and-drop operation. In order to give a greater level of generalization, i.e. run with different datasets that have the same data structure, a model can be set both input and output Model Parameters.

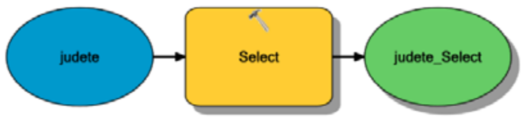


Figure 3. A Model that uses the Select by Attribute tool

Along with setting model parameters, users can also define independent variables which behave as parameters and can be used by the tools from the model.

In a model, there can be more than one thread of operations that run in parallel. In case the operations from a certain thread depend on the output of another one, the user can include preconditions in order to make sure the necessary steps have been completed prior to that specific operation. Preconditions are added using the *Connect* button from the ModelBuilder toolbar and selecting the *Precondition* option from the pop-up menu.

Threads that work with the same dataset are called *branches*. In a workflow with multiple branches, in order to reduce the visual complexity of the model's diagram, ModelBuilder provides a specific *Merge Branch* tool which can be added to merge the multiple branches in a single one. A *Stop* tool is also present in order to be able to stop the execution of a model under specified conditions.

In their goal to be as convenient as possible for the users who are not programming specialists, ESRI featured *ModelBuilder* with iterative tools that share a similar behavior with acknowledged programming languages such as the *For* or *While* iterators. However, their practicality is rather limited compared to structured programming. For instance, the variable used in the *For* iterator is automatically incremented without having the possibility to alter it in the different logic. Another limitation is that only one iterator can be used in a model, regardless of its type.

Also, the conditional operations such as *If / Else*, *Switch* or *Case* are not included as tools in ModelBuilder. This can be overcome by the possibility of including Python scripts within the models.

Along with a built-in Python compiler, ArcGIS provides a library of Python scripts – *arcpy* – for virtually any geoprocessing and data management instrument available in *ArcToolbox*. Including and using a Python script in a model is similar to using a tool. Scripts can have input parameters (Model Parameters or outputs from previous operations) and can return results that can be used as input data for other tools or scripts in the model.

ArcGIS is not the single GIS software providing a geoprocessing modelling set of features. Although modules such as AutoCAD Map 3D Workflow Designer, QGIS Graphical Modeler or Edras Imagine Spatial Model Editor [1] offer some possibilities to batch geoprocessing tasks, they are not subject for the current study.

5. Case Study - urban population distribution analysis by county

In order to showcase the manner a software component is developed using the ModelBuilder feature, starting from a threshold as an input parameter, a geoprocessing model was built to generate a classification consisting of 3 categories for Romania’s counties:

- The class of counties which do not have population in large cities (cities with more than 100.000 inhabitants);
- The class of counties where the population in large cities’ share in the total of Romania’s population in large cities stands for less than the input threshold;
- The class of counties with the referred share above the input threshold.

The Model takes as input data the following elements: the population table – *populatie* - which holds data on the population of counties, the spatial feature for counties – *judete* – the name of the field which will store the computed class for each county and which is going to be added to the counties feature – *Nume_Camp* – and the threshold which is also a Model Parameter - *Prag*. The execution of the model will establish the class for each county and apply a map colour scheme [4] using a *Unique Values Symbology* imported from an ArcGIS specific symbolization Layer file - *lyr* – which was previously generated.

The model was built using tools from ArcToolbox and Python scripts connected in the processing flow through a suitable set of parameters (both input and output) as well control elements for the workflow, specifically the employment of a pseudo-alternative structure.

Combining visual elements with Python scripts enables a greater flexibility when defining the workflow since not all processing is available using solely tools from ArcToolbox.

The model's diagram is depicted in *Figure 4*.

In order to be able to execute the model multiple times, along with the desired computation, the model was designed to clean the workspace for any temporary variables and fields such as the absolute value of the share of each count in the Romania's total large cities population. Also, the model verifies if the cluster field has already been added from previously executions of the model or if it is required to be added to the counties feature. However, as stated before, conditional operations are not supported by ModelBuilder through regular uses of the ArcToolbox library. Therefore, to achieve the desired check, a custom Python script – *Este_camp.py*- was added to the flow. The script, detailed in *Code section 1*, receives the field's name as text and returns a Boolean value. The *False* value is a necessary Precondition for the *Add_field* tool.

Along with the ModelBuilder's *Merge Branch* tool, this pseudo alternative structure enables having a unitary subsequent set of operations.

```
import arcpy
num_c = arcpy.GetParameterAsText(0)
nume_camp_col =
      [camp.name for camp in
arcpy.ListFields("judete")]
if num_c in nume_camp_col:
      arcpy.SetParameter(1, True)
      arcpy.SetParameter(2, False)
```

```
    else:
          arcpy.SetParameter(1, False)
          arcpy.SetParameter(2, True)
```

Code section 1. Este_camp.py - checks if a field exists within a table view

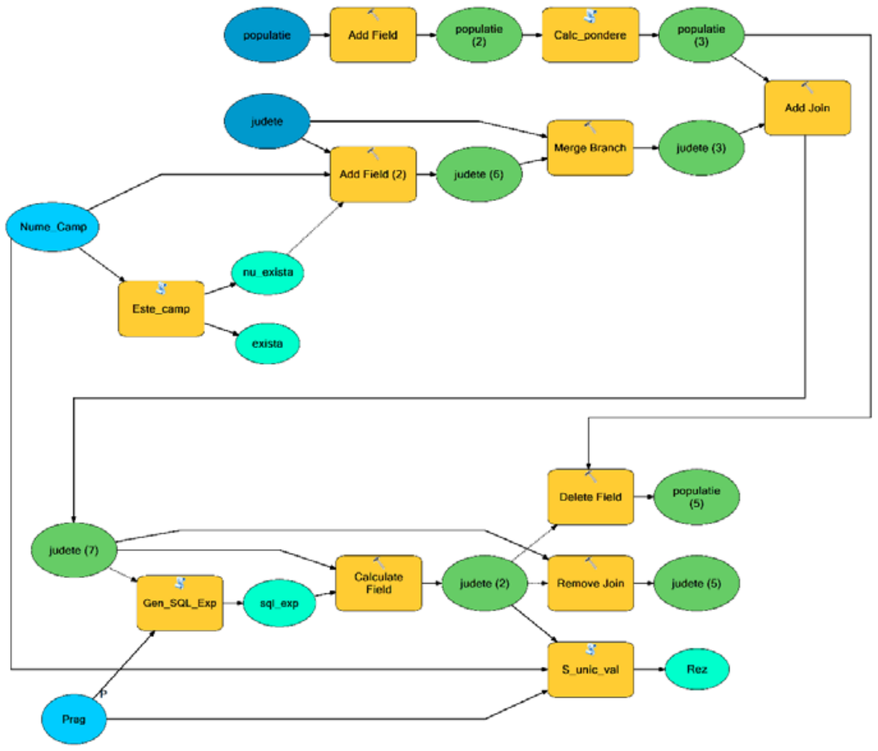
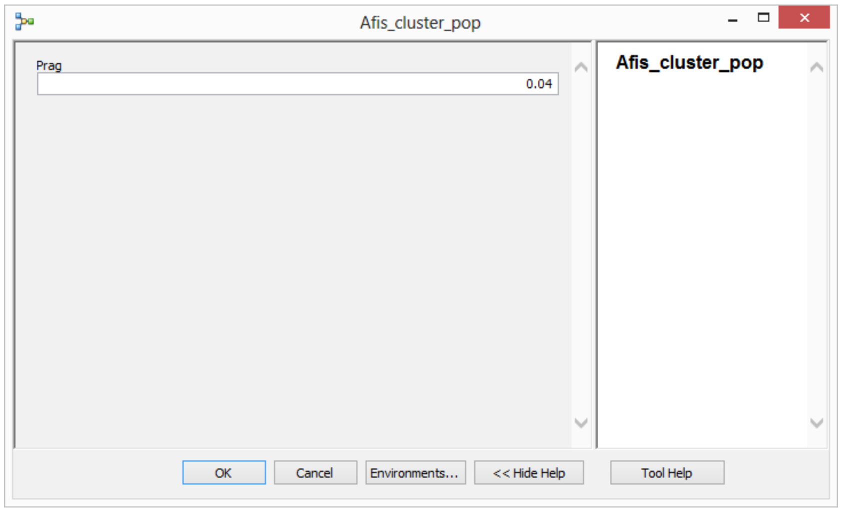


Figure 4. The Case Study model's diagram

In Python, using the *arcpy* library and the Data Access Module - *da*- users can browse and update a table view which is set as an input parameter. The *Calc_pondere.py* script, detailed in *Code Section 2*, is used to compute the share based on which the intended cluster is created, and outputs the updated population table.

```
import arcpy
import arcpy.da as da
tab = arcpy.GetParameter(0)
rd_col = [randuri for randuri in da.SearchCursor(tab,("cp"))]
s=0
for val in rd_col:
      s+=val[0]
with da.UpdateCursor(tab,("cp","pondere_cp")) as crs:
      for rd in crs:
            rd[1] = float(rd[0])/s
            crs.updateRow(rd)
arcpy.SetParameter(1,tab)
```

Code section 2. Calc_pondere.py – calculates the share value of each county's large cities population in Romania's total



Prag
0.04

Figure 5. The model's interaction window – a standard User Interface

Although ArcToolbox features a *Calculate_field* instrument for updating table views, which also allows the execution of a Python code block, there are limitations in working with various parameters, in our case, the threshold Model Parameter. To overcome this, as shown in *Code Section 3*, a custom SQL expression is built, based on the threshold, and set as the input parameter for the instrument to call the code block – *Code Section 4*, which updates the counties attribute table with the required cluster value.

```
import arcpy
s_prag = arcpy.GetParameterAsText(0)
var_s = "get_clust( !populatie.pondere_cp! ," + s_prag + ")"
arcpy.SetParameterAsText(1, var_s)
```

Code section 3. Gen_SQL_exp.py – generates input parameter for the Calculate_field tool

```
def get_clust(ppop, prag):
    if ppop== 0:
        return 1
    elif ppop < prag:
        return 2
    else:
        return 3
```

Code section 4. Code block to be executed by the *Calculate_field* tool

In order to apply a colour scheme to the document map according to the cluster another script is added to the model – *Code Section 5*. Compared to manually applying the symbology using the GUI, the script has several limitations out of which, relevant to our study, is the impossibility of setting a custom colour scheme without having to use a previously generated Layer file.

```
import arcpy
num_jud = arcpy.GetParameterAsText(0)
num_c = arcpy.GetParameterAsText(1)
p_val = arcpy.GetParameter(2)
lim_val = p_val * 100
mxd = arcpy.mapping.MapDocument("CURRENT")
df = arcpy.mapping.ListDataFrames(mxd, "Romania")[0]
jud_l = arcpy.mapping.ListLayers(mxd, num_jud)[0]
l_sursa = "d:/lucru/simb_uv_pol.lyr"
layerSymb = arcpy.mapping.Layer(l_sursa)
arcpy.mapping.UpdateLayer(df, jud_l, layerSymb, "TRUE")
if jud_l.symbologyType == "UNIQUE_VALUES":
      sir1 = "sub " + str(lim_val) +"%"
      sir2 = "peste " + str(lim_val) +"%"
      jud_l.symbology.valueField = num_c
      jud_l.symbology.addAllValues()
      jud_l.symbology.showOtherValues = False
      jud_l.symbology.classLabels = ["0% in orase mari",
sir1, sir2]
      arcpy.SetParameter(3, True)
```

```
else:
      arcpy.SetParameter(3, False)

arcpy.RefreshActiveView()
arcpy.RefreshTOC()
```

Code section 5. *S_unic_val.py* – Applies the Unique Values Symbology over the document map

The model interracts with the user through a Standard User Interface as shown in *Figure 5*. Successfully running the model returns the *Rez* variable with a *True* value and updates the map document's Active View and Table of Contents as in *Figure 6*.

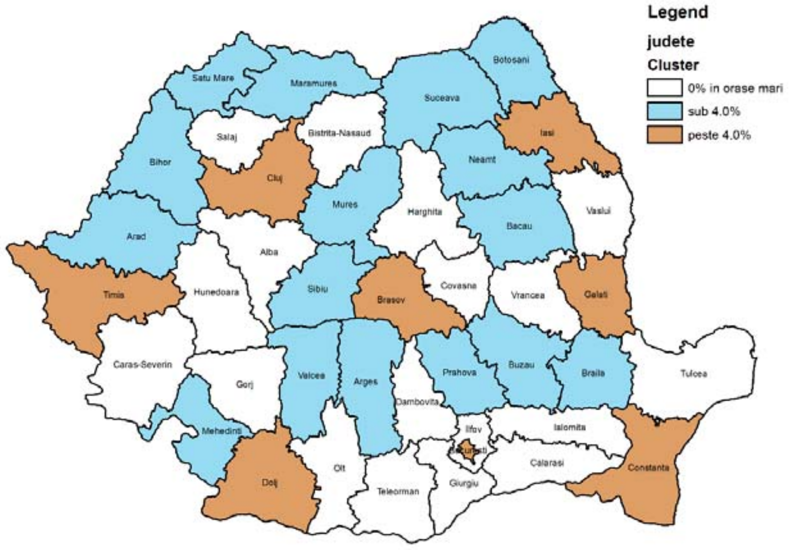


Figure 6. The computed map with a 4% threshold as input for the Model Parameter

6. Conclusions

Visual tools for software development are easier to use and more intuitive for the users who are not programming specialists. Simultaneously, they do offer the possibility to include code sequences which confer a greater flexibility and efficiency for processing. However, these tools have limitations which include the standard user interface, the use of predefined instruments, a restrictive working context and limited processing complexity.

The showcased component, ModelBuilder, is designed using similar concepts and principles as other acknowledged modeling tools for software development such as *executable workflow diagrams* and *forward engineering* options that result in a 100% complete code generation regardless of the model's complexity.

References

[1] Dobesova, Z., Data Flow Diagrams in Geographic Information Systems: a Survey, Conference Proceedings of The 14th International Multidisciplinary Scientific GeoConference SGEM 2014, ISBN 978-671-7105-10-0, pg 541-548

[2] Nabil, A., *Towards E-CASE Tools for Software Engineering*, International Journal of Advanced Corporate Learning (iJAC). ISSN: 1867-5565, Vol 6, No 1 2013, pg. 16-19;

[3] Petre, M., *UML in practice*, ICSE '13 Proceedings of the 2013 International Conference on Software Engineering, IEEE Press Piscataway, NJ, USA, 2013, pg. 722-731;

[4] Reveiu, A., Dârdala, M., *Techniques for Statistical Data Visualization in GIS*, Informatica Economica, vol. XV, nr. 3, Editura INFOREC, ISSN 1453-1305, Bucuresti, 2011, pag 72-79;

[5] Rumbaugh, J., Jacobson, I., Booch, G., *The Unified Modeling Language Reference Manual*, Addison Wesley Longman Inc., ISBN 0-201-30998-X, Massachusetts, 1999, pg. 3-38;

[6] Steiniger, S., Hunter, A., *The 2012 free and open source GIS software map – A guide to facilitate research, development, and adoption*, Computers, Environment and Urban Systems, Volume 39, Elsevier Ltd, May 2013, Pages 136–150

[7] Stephen J. Mellor, S.J., Balcer, M., *Executable UML: A Foundation for Model-Driven Architectures*, Addison-Wesley Longman Publishing Co., Inc., ISBN:0201748045, Boston, MA, USA, 2002;

[8] Smeureanu, I., Drula, G., *Multimedia concepte si practica*, CISON, Bucuresti 1997

[9] ARC Advisory Group, Geographic Information System Global Market Research Study. http://www.arcweb.com/market-studies/pages/geographic-information-systems.aspx. Retrieved March 2016

[10] Environmental Systems Research Institute, Inc., *Executing tools in ModelBuilder*, http://desktop.arcgis.com/en/arcmap/10.3/analyze/executing-tools/executing-tools-in-modelbuilder-tutorial.htm , 2016

[11] Oracle Corporation, *MySQL™ Workbench Reference Manual*, revision: 47148, 2016 - http://dev.mysql.com/doc/workbench/en/