[PD-03-028] Visual Programming for GIS Applications

Abstract

Visual programming languages (VPLs) in GIS applications are used to design the automatic processing of spatial data in an easy visual form. The resulted visual workflow is useful when the same processing steps need to be repeated on different spatial data (e.g. other areas, another period). In the case of visual programming languages, simple graphical symbols represent spatial operations implemented in GIS software (tools, geoalgorithms). Users can create a sequence of operation in a simple visual form, like a chain of graphical symbols. Visual programs can be stored and reused. The graphical form is useful to non-programmers who are not familiar with a textual programming language, as is the case with many professionals such as urban planners, facility managers, ecologists and other users of GIS. VPLs are implemented not only in GIS applications but also in remote sensing (RS) applications. Sometimes both types of applications are bundled together in one geospatial application that offers geoalgorithms in a shared VPL environment. Visual programming languages are an integral part of software engineering (SE). Data flow and workflow diagrams are one of the oldest graphical representations in informatics.

Keywords: automate, Python, software engineering, visual communication, VPL

Author & citation

Dobesova, Z. (2020). Visual Programming for GIS Applications. The Geographic Information Science & Technology Body of Knowledge (1st Quarter 2020 Edition), John P. Wilson (ed.). DOI: 10.22224/gistbok/2020.1.7.

Explanation

1. Definitions
2. Advantages and Disadvantages of Visual Programming Languages (VPLs)
3. History of VPLs in GIS and Remote Sensing Applications
4. Overview of VPLs
5. Evaluation of Effective Cognition through VPLs

1. Definitions

A **Visual Programming Language (VPL)** is a programming language that uses graphical symbols and graphical connectors for the design of algorithms. In the case of GIS, the algorithms are aimed at processing spatial data. The opposite of visual programming languages is textual programming languages. Both visual and textual languages have **syntax** and **semantics**. **Textual programming language** use sequences of characters. The sequences are legal and illegal strings. The syntactically valid programs are constructed from the concrete characters or strings of characters, for example, keywords such as define, if, for, or class. Keywords are words that are "reserved from use" in a programming language. The keyword cannot be used as an identifier, such as the name of

a variable, function, or label. The character sequences for identifiers are user suggestions that belong only to some conventions or limitations. New users must memorize a list of keywords in textual language and acquire the syntax rules for the writing of a program.

Terminology of VPL:

- **Visual language** (or visual notation, graphical notation, diagramming notation): a set of graphical symbols (visual vocabulary) and associated visual syntax and semantics.
- **Visual vocabulary**: a set of **graphical symbols** (elements and connectors). Elements are **nodes** in a diagram and connectors connect the nodes.
- **Visual semantics:** a set of definitions of the meaning of each graphical symbol.
- **Visual syntax** (visual grammar): defines the formal structure of the program. It a set of compositional rules on how to form and join symbols from visual vocabulary together.
- Graphical symbols are used to symbolize (represent visually) **semantic constructs**, typically defined by a meta-model. The meanings of graphical symbols are defined by mapping them to the constructs they represent. Examples of semantic constructs are “point data” or “spatial operation buffer,” and both have their graphical representations.
- A valid expression in visual notation is called a **visual sentence** or **diagram**. Diagrams are composed of symbol instances (tokens), arranged according to the rules of visual grammar. GIS and RS software use the following terms for a visual sentence: **Geoprocessing Model**, **Diagram,** and **Workflow**. These three terms can be assumed to be synonyms in the context of VPLs for GIS.

VPLs provide the protocols not only for the design of the algorithm as a sequence of operations but also allow the user to interact with the system itself and thus ultimately represent separate executable models.

2. Advantages and Disadvantages of Visual Programming Languages (VPL)

Advantages of the graphical form include ease of design (very often through drag and drop functionality), readability, comprehension and reuse. The graphical form helps users to clearly envision an algorithm in the same way that CASE tools do for the graphical design of database structures. Visual programming languages automate tedious and repetitive tasks and allow the user to run long workflows faster instead of manually running isolated operations. The workflow could become a batch process. Moreover, a screenshot of the diagram can be used to document and archive the steps of the workflow. Visual programming has a high self-reported ability, which means that other comments or texts may not be necessary to explain the steps of a workflow: sometimes, the picture of the workflow is enough.

According to Hexagon Geospatial (2015) , the advantages of visual programming languages include:

- Experts create the process once, and others can re-use it.
- Workflows can be distributed to non-experts.
- The prepared workflows save time, money and resources.
- Processing data with the same workflows introduces standardization and consistency.

The main advantages of VPL in comparison with textual programming are:

- VPL is more intuitive than textual programming.
- VPL is easier to learn than textual programming.
- VPL as a starting point for advanced textual programming.

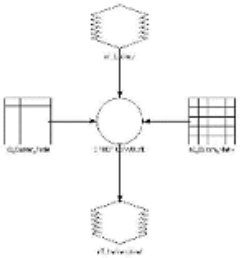
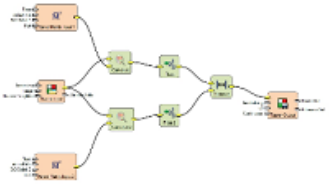
From the perspective of the learning process, visual programming could be a valuable preliminary step for starting to learn textual programming. It is possible to convert most existing workflow diagrams to a functional textual program such as Python (available in ArcGIS ModelBuilder, QGIS Processing Modeler and GRASS GIS Graphical Modeler).

An overview of visual programming languages is provided below (Table 1). The table contains basic information about visual vocabularies (symbols and connectors), functionalities, and available programming constructs. Parametrization has to do with the status of data. In some languages, it is only possible to design workflow for existing, specific data (stored on the disk). The workflow is invariantly tied to concrete data; there is no possibility to change input data in various utilizations. When parametrization of workflow is possible (set for some inputs – data or variables), inputs do not have to be assigned to specific data until the moment a workflow is run. When the workflow is started the input data could the freely chosen. The parametrization and parameters are depicted in various forms in VPLs. QGIS Processing Model is the best, from that perspective, as the model is designed as a parametrical workflow from the start in all cases. So QGIS does not need any depiction what is a parameter. Otherwise, ArcGIS ModelBuilder allows users to switch between parametrical and non-parametrical inputs, something that is depicted by the letter P near the symbol within the ModelBuilder interface.

The actual shape of connectors is relevant because the interpretation of curved connectors is sometimes made more difficult due to overlaps and crossing, especially when no auto-align function is available. This overview is based primarily on documentation and on the practical experience of the author in workflow design. Greater details of these VPLs as well as others are provided below in section 4.

Table 1. Overview of Visual Programming Languages used with GIS

Software	Name of VPL	Symbol Shape	Symbol Color	Parameterization	Connectors	Preferred Orientation	Inserting Sub-Models	Loops and Conditions	Disable Operation from a Flow
ArcGIS	Model-Builder	various shapes	various colors	Y	linear	H (V)	Y	Iterator If-Then-Else (ArcGIS Pro)	N
		ArcGIS 10.x 				ArcGIS Pro 			
IDRISI	Macro Modeler	rectangle and rhombus	various colors	N	linear	H	N	N	N
									

ERDAS IMAGINE	**Model Maker**	various shapes	b/w	N	linear	V	N	N	N
									
	Spatial Model Editor	rectangle	4 various colors, big inner icon	Y	curved	H	Y	Y	N
									
ENVI	**ENVI Modeler**	rectangle	various colors	Y	linear, curved	H	Y	Iterator	N
									
AutoCAD MAP 3D	**Workflow Designer**	composite rectangle symbols	gray, green START, red END	Y	linear	only V	Y	Y	Y
									
QGIS	**Processing Modeler**	rectangle	3 colors	Y default	curved	D (H/V)	Y	N	Y
		 version 2  version 3							
GRASS GIS	**Graphical Modeler**	various shapes	various colors	Y	linear	V/H	Y	Y	N
									

Notes: Y = Yes; N = No; V = Vertical (top to down); H = Horizontal (left to right); D = Diagonal.

In spite of their benefits, the disadvantages of VPLs need to be recognized. VPLs have limitations, such as the implementation of loops. Some constructs are not fully implemented in VPLs like cycle FOR, condition WHILE, or switch SELECT CASE. They are generally not available in VPLs and only appear in the newest versions of some (e.g. ModelBuilder in ArcGIS Pro 2.x has a new If-Then-Else tool). Cycle controls like Iterator are partly implemented in ModelBuilder and ENVI Modeler, but not as a pure cycle for the repetition of a set of commands. And, there are further limitations. In ModelBuilder, only one iterator per model is allowed. Some operations are only accessible as programming methods in textual programming. In ModelBuilder for instance, it is hard to express the order of processes in VPL, so the construction Precondition is available. From a textual imperative programming perspective, it is a strange construct.

Some positive features of VPLs have become disputable in case of huge diagrams and complex processing (the good readability, modularizations - dividing to the chunks as procedures, grouping). Sometimes it is difficult to find the start and the end of the workflow diagram. Only AutoCAD Map Workflow Designer has symbols for START and END.

VPLs have limitations, and some tasks require switching to a textual language as the necessary next step. Debugging tools and license management (for some functions) are typically not accessible through VPLs. In these use cases, switching to textual programming like Python scripting is generally necessary. Experienced users end up using only textual programming in practice, in case of frequent demand for various batch processing of spatial data. Further discussion about the suitability of visual and textual programming for different types of use cases is an interesting topic, but beyond the scope of this entry.

3. History of VPLs in GIS and Remote Sensing Applications

This short chronological overview presents the history of VPLs from the first release of a VPL to the latest implementations in the area of GIS and RS (but the present situation is evolving rapidly). ERDAS IMAGINE was one of the first commercial software packages to offer a graphical geospatial data and workflow modeling tool: Model Maker was introduced in 1993 as a graphic flow chart model building editor (ERDAS 1993). Then Macro Modeler editor appeared in the IDRISI 32 Release 2 software from Clark Labs in 2001, to create workflow diagrams. ESRI followed and released the ModelBuilder component in 2004 for its software ArcGIS, version 9. Workflow Designer appeared in 2009 in AutoCAD Map 3D, produced by Autodesk (Dobesova 2014). ERDAS IMAGINE v. 2013 has a new component for VPL named Spatial Model Editor. Both editors (Model Maker and Spatial Model Editor) are present simultaneously in the interface, and it is possible to use either according to user wish. It is possible to convert existing models from the older to the newer editor.

In 2013, two new graphics editors were released: Processing Modeler for QGIS (2.0 Dufour) and Graphical Modeler for GRASS GIS; these are both open source GIS software programs. QGIS Processing Modeler was remade and improved in version 3.0 in 2018. Remote sensing software ENVI, by Harris Geospatial Solutions, introduced a new VPL, ENVI Modeler, in version 5.5 in 2018.

Other VPLs exist in the form of plugins for Rhino3D Grasshopper 3D (Bison, Groundhog,

Docofossor). In the area of ETL processes (Extract, Transform, Load) for processing data in warehouses, software FME and GeoKattle are available.

Note that VPLs also exist for many purposes other than GIS and RS. Application areas are notably education, multimedia, video games, simulations, data warehouse, business intelligence, data mining, civil engineering, robotics etc. A dynamical list of near hundred VPL applications can be found at https://en.wikipedia.org/wiki/Visual_programming_language.

4. Specifics about VPLs for the Geospatial Sciences

Section 4 describes each VPL in detail to present the visual languages and their respective vocabularies, including how the vocabulary differs in not only shapes and colors of basic symbols (semantics) but also significantly in syntax and amount of functionality. The main differentiation in syntax is the interleaving (or no interleaving) of geoprocessing symbols with data symbols. ModelBuilder is the typical syntax representation of processes and data symbols interleaving. Otherwise, QGIS's Processing Modeler represents groups of processes that are connected directly. Only the final data could have a symbol in a workflow. In the case of AutoCAD Map Workflow Designer, the syntax only allows the connection of operations.

The overview starts with commercial software followed by open-source software. The software could also be divided into GIS and RS groups. But in some cases, one software package offers to use geoalgorithms both for vector and raster data in its visual language. For example, ERDAS IMAGINE is RS software, but it is possible to use vector operations adopted from Geomedia in its Spatial Model Editor. QGIS Processing Modeler also offers a large number of algorithms for both raster processing (from SAGA, GRASS GIS) and vector processing (from QGIS and GRASS GIS).

4.1 ModelBuilder for ArcGIS Desktop

ModelBuilder and geoprocessing models are very commonly used by GIS practitioners. ModelBuilder is a typical VPL and may be the most well-known VPL implementation in GIS. Figure 1 below shows the interface of ModelBuilder and a model example in ArcMap v. 10.x. The graphical vocabulary consists of the blue ovals for input data and green oval for derived/result data. Processes (tools) are represented by yellow rectangles. In addition, the graphic symbol Iterator (orange hexagon) represents repetition or cycles (Dobesova 2013b). The main connectors are black lines with arrows that represent the flow of data. Additional connectors exist (e.g. precondition, environment and feedback). Users can change the symbol labels. Moreover, it is possible to set the input data or variables as parameters of the model. A parameter is expressed by the letter P near the oval symbol. It allows the user to change the input data and to reuse the model on different data. Models are stored in named custom toolbox, *.tbx files, that can be simply copied for others to use. It is also possible to insert a nested model – another existing model into the main model (like a subroutine). Moreover, Python scripts can also be inserted into a model.

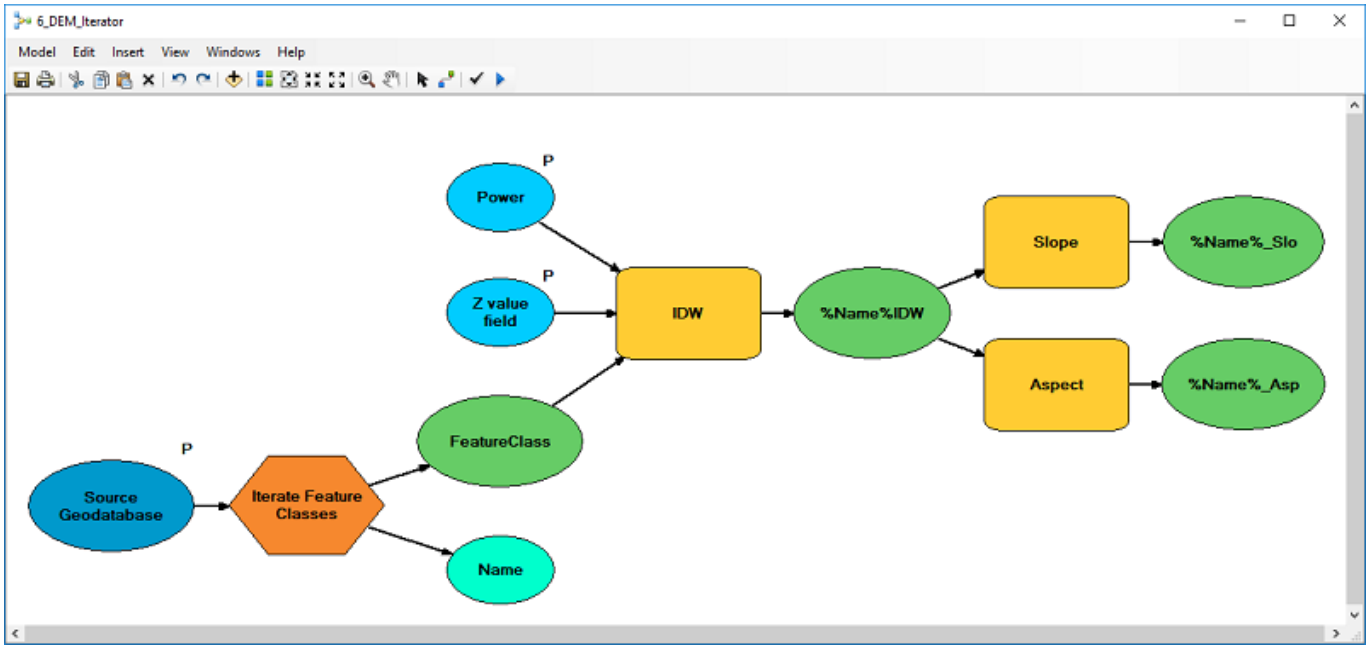


Figure 1. Example of the model in ModelBuilder for ArcGIS v. 10.x. Source: author.

ModelBuilder is also available in ArcGIS Pro v. 2.x. Graphical symbols have slightly different (lighter) colors (Figure 2). One new functionality is the ability to group symbols as a subset. New logical IF tools have also been introduced. Ten various IF conditions test the existence of data, attribute data type etc. The symbol of logical element IF is a yellow rhombus. Both ModelBuilder versions allow to automatically convert models to Python scripts.

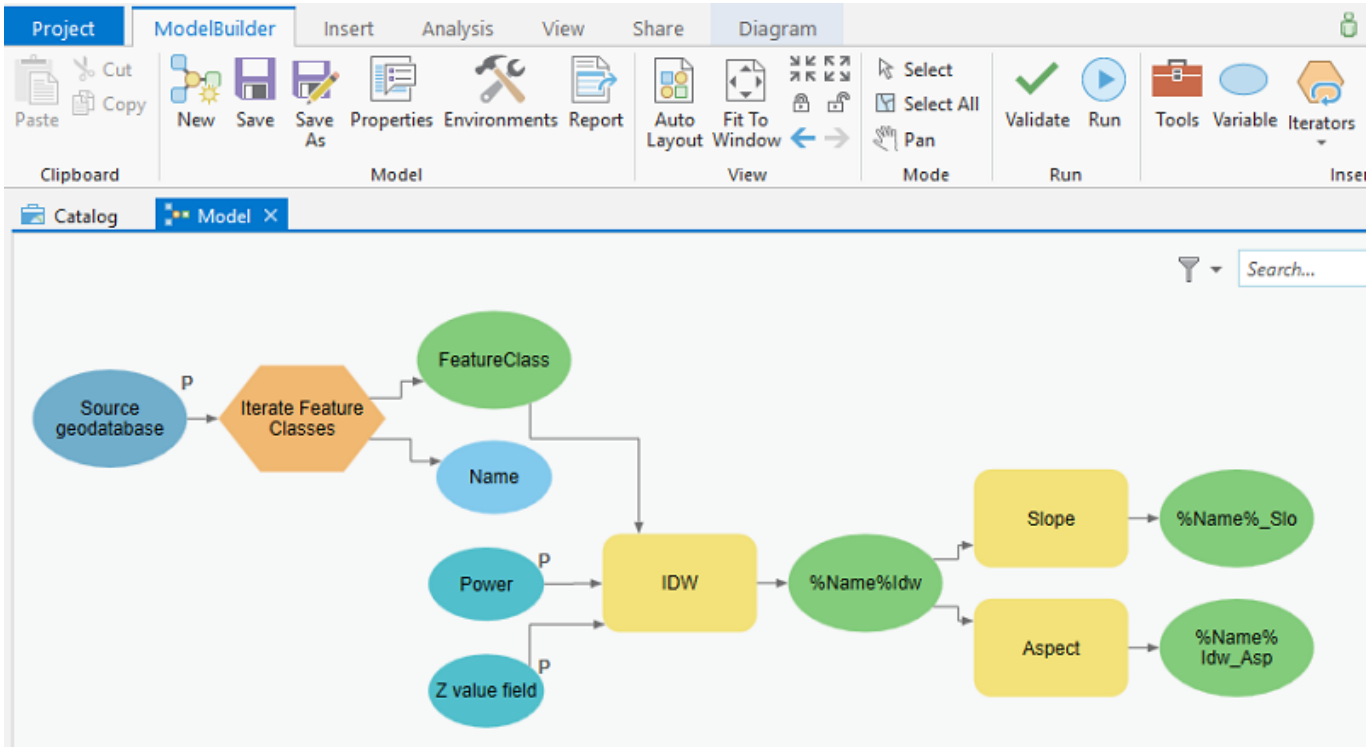


Figure 2. Example of a model in ModelBuilder for ArcGIS Pro v. 2.x (the same model as in Fig.1). Source: author.

4.2 Macro Modeler for IDRISI

The graphical vocabulary in Clark Lab's IDRISI Macro Modeler consists of simple sharp rectangles for data and rhombus for operations. The rhombus has a pink color fill. The color fill of rectangles differs with the type of data: violet is a raster, green is a vector, yellow is an attribute (Figure 3). Additionally, there are rectangles for dynamic groups (red labels) and group files (not shown in Figure 4). The element size is the same for all symbols, but text labels differ in thickness. Labels for operations are in boldface, while others have a normal face. The background of the model is yellow.

The connecting lines are straight or right-angled dark blue lines with an arrow at the end. An uncommon functionality is the possibility to construct processes with feedback in this modeler. Feedback lines are red. The orientation of flow is variable in the editor, but left to right is preferable. It is possible to insert a submodel. Models are stored in *.imm files. TerrSet, as the latest version of IDRISI, has incorporated Spatial Decision Modeler as a graphical modeling tool for multi-criteria and multi-objective decision support.

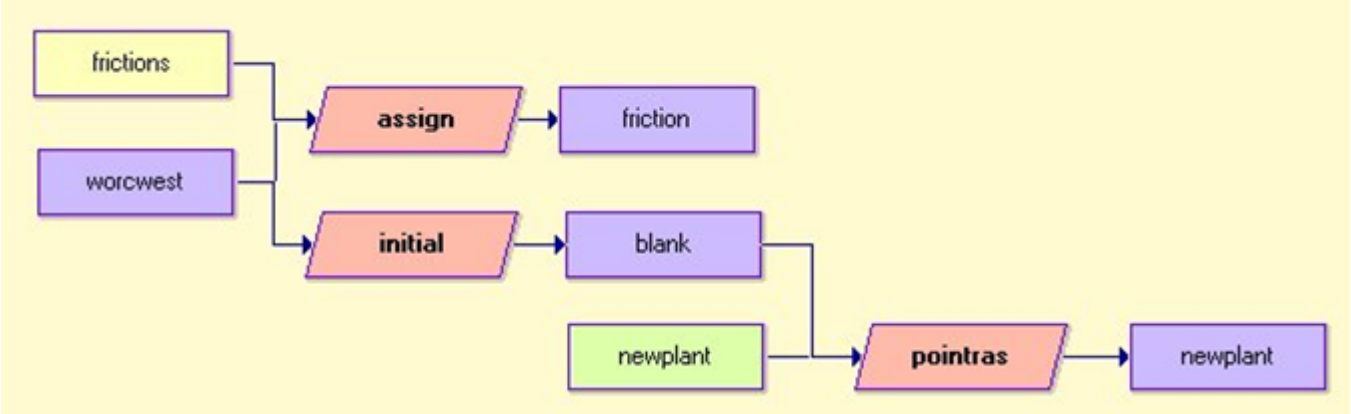


Figure 3. Example of a model in Macro Modeler for IDRISI. Source: author.

4.3 Model Maker and Spatial Model Editor for ERDAS IMAGINE

Both the older editor Model Maker and the newer editor, Spatial Model Editor, are currently available in Erdas Imagine. Model Maker uses only black and white graphical symbols. The shapes of symbols express different types of data effectively, by suggesting their meaning. The shape for raster data resembles a set of bands, the symbol for tables and matrices is a rectangle with one or more columns and rows. Functions are expressed by a circle (Dobesova 2014), the universal symbol for functions. Connectors are solid straight arrows (Figure 4). Flow orientation is variable: users can construct a top-down oriented model, a left-right oriented model or mixture of both. Automatic alignment is not available. Symbol labels have a fixed position and are not changeable. The model is saved in a structured text file with a * .gmd extension.

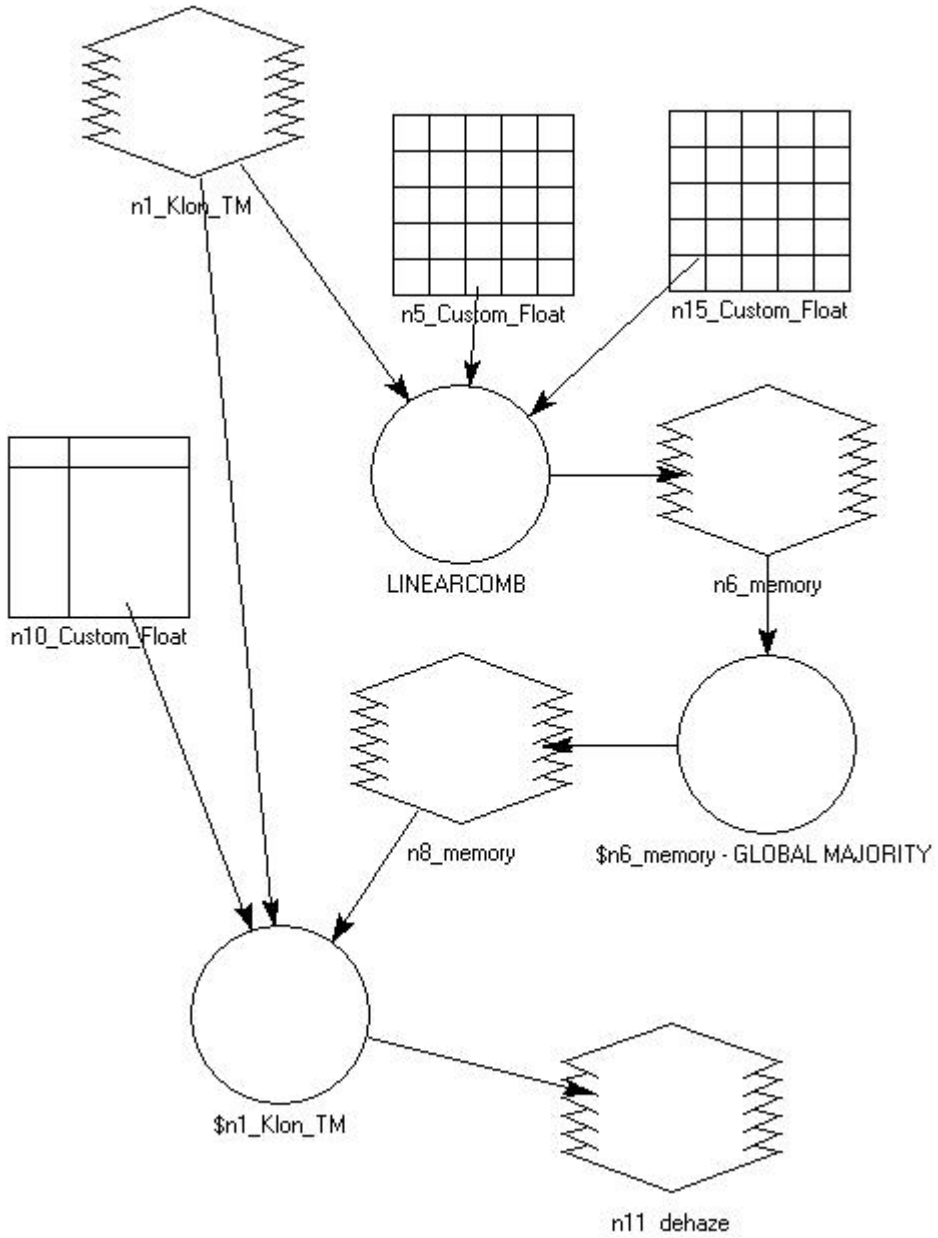


Figure 4. Example of a model in Model Maker for Erdas Imagine. Source: author.

Spatial Model Editor has a totally different notation. Symbols have colors and a big inner icon. All symbols are rounded rectangles with a color fill (Figure 5). The pink color is for data (input/output raster, scalar, table, vector, etc.).The light green fill is for operators and operations. Each rectangle has ports, small arrows where lines connect. The color fill of arrows depends on setting status: red fill for an unset value, gray for a set value, yellow for output/derived values. Each rounded rectangle has an inner color icon that expresses the type of data or type of operation (e.g. icon π means scalar). The icons are visually meaningful (evocative) and help to explain the diagram. Each symbol has an inner label that is added automatically – e.g. the name of the operation. Users can rename the labels to better express specific values or operations. The size of rectangles varies and depends on the number of ports (they can be added) or on the length of the inner text labels (Dobesova 2014). Sub-models can be incorporated into the main model. The brown sub-model symbol can be double-clicked to open it directly from the main model. The preview

symbol allows to immediately view the result data at the end of a chain of operations. Models are stored in an XML file with a * .gmdx extension.

Old models from Model Maker can be converted automatically to the new Spatial Model Editor format.

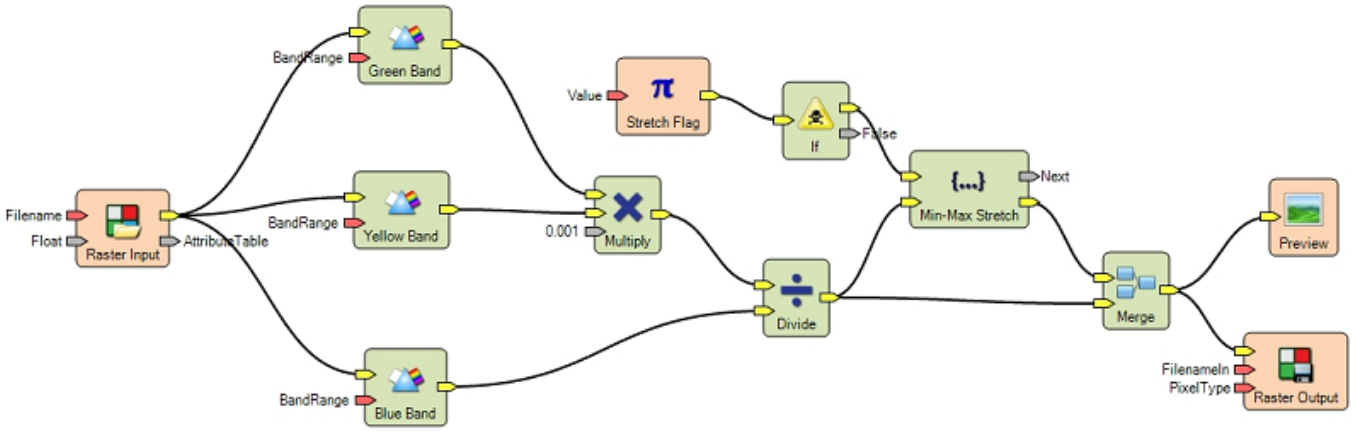


Figure 5. Example of a model in Spatial Model Editor for Erdas Imagine. Source: author.

4.4 ENVI Modeler

ENVI, a software application for the processing of remote sensing data, also has a graphical editor named ENVI Modeler. All graphical symbols (nodes) are rounded rectangles. The symbols are distinguished by color. Operations are expressed by yellow rectangles. The connectors are straight or curved lines (Figure 6). The ENVI 5.5.2 version has a new orange graphical element, Filter Iterator, that executes an operation only when a specified condition is met. This operation iterates through a collection of data while also applying the specified condition. Multiple Aggregator nodes can be combined; for example, using one Aggregator node to close a loop introduced by an Iterator node, and using another Aggregator node to collect items into an array. The resulting data can be viewed in a model by connecting them to a View node.

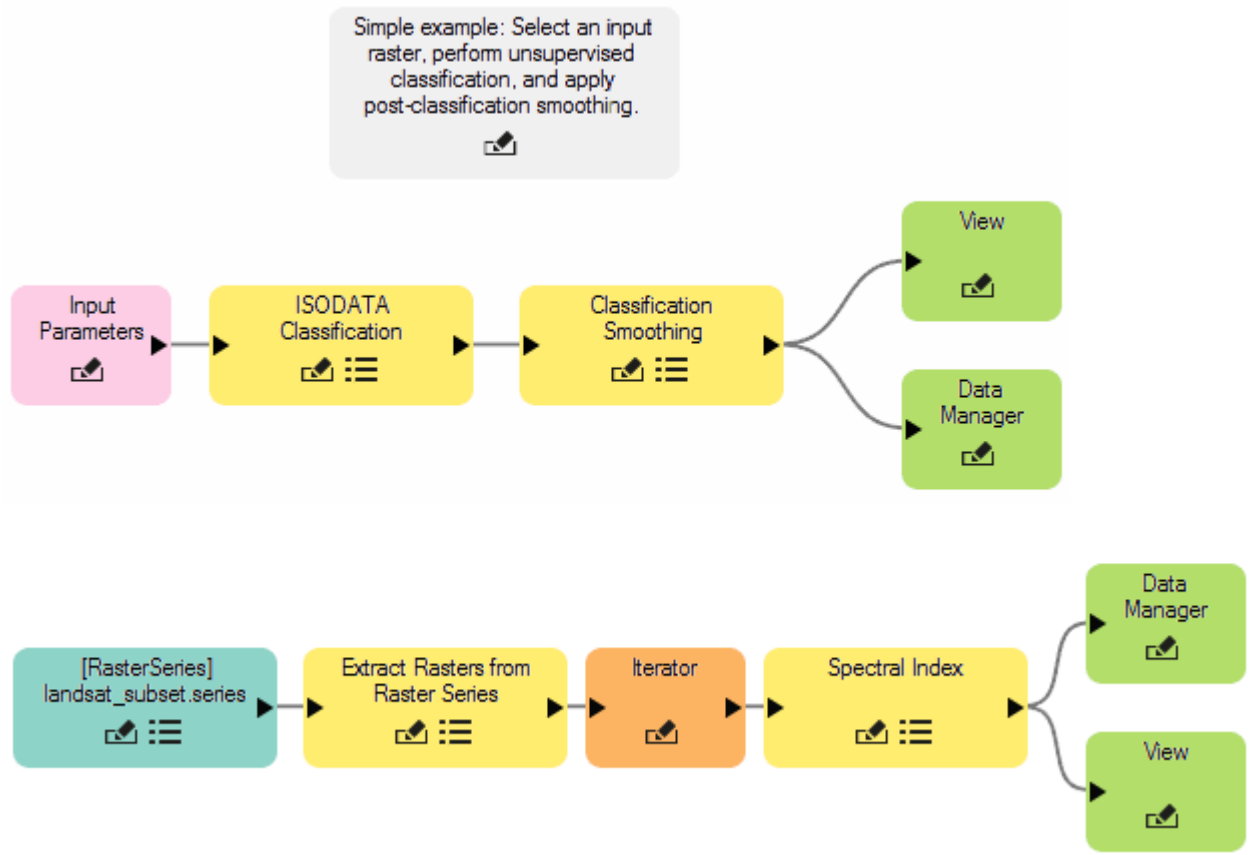
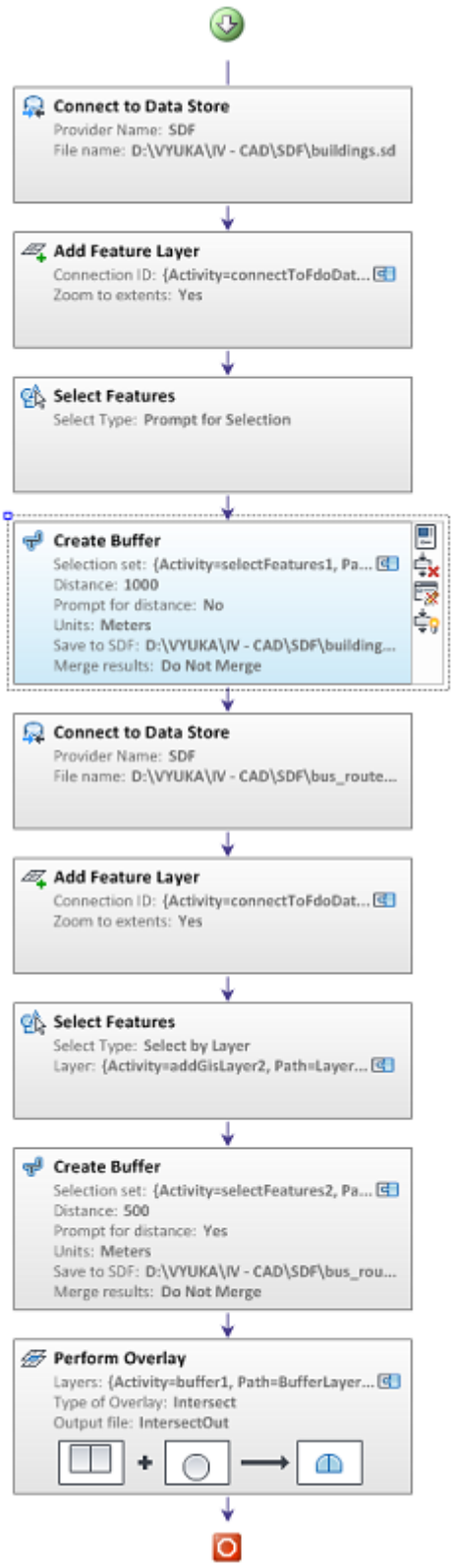


Figure 6. Two examples of models from ENVI Modeler. Source: L3HARRIS.

4.5 Workflow Designer for AutoCAD Map 3D



Workflow Designer is based on technology by the Microsoft Windows Workflow Foundation. The graphical symbols are big gray rectangles. There are no distinct symbols for operations and data, as opposed to the graphical dictionaries of previously described VPLs. Workflow designer uses composite graphical symbols. The gray rectangle expresses operations with settings, input data, parameters and output data. Connectors are very short black lines that are added automatically to the diagram. The orientation of the diagram is fixed to top-down but it is possible to to zoom in/out of the diagram to see details. The diagram starts with a green arrow symbol and ends with a red stop symbol (Figure 7).

The text in the rectangle conveys a set of information. The first bold line is the name of the operation. The next lines are the names and values of the set parameters. The first line also features, in the upper left corner, a small colored icon that indicates the type of operation. For analytical and overlay operations, the entire process is illustrated by a graph at the bottom of the rectangle (Figure 7 - last operation Perform Overlay). Operations are set in detail through functional icons that pop-up on the right side of the rectangle. The functional icons are visible in Figure 7 for the Create Buffer operation, which is selected and appears

in a blue rectangle. The functional icons on the right side are visible only after selection.

Individual operations can be excluded from processing. The rectangle symbol is then covered by a light green rectangle and does not need to be deleted from the diagram.

The workflow is stored in a structured text file with the extension * .xoml.

Figure 7. Workflow diagram in AutoCAD Map 3D. Source: author.

4.6 Processing Modeler (graphical modeler) in QGIS

There are two implementations of VPL in Open Source GIS. The first is the Processing Modeler for QGIS, also called the graphical modeler, and authored by Victor Olaya, and the second is the Graphical Modeler for GRASS GIS. Processing Modeler in QGIS uses only three graphical symbols – all rectangles with the same dimensions. The rectangles differ by color fill. The fill for input data is blue. The process has a white fill and output data a light blue (turquoise) fill (Figure 8). The colors have changed in the new version 3.x: input data are yellow and output data green (Figure 9). Operations remain in white. Processing Modeler offers operations from other software and libraries. A mix of them can be used in the same model. The symbol for operations contains a small icon on the left side indicating which library the tool comes from (QGIS, GDA / ORGL, SAGA, Orfeo, GRASS, etc.). Processing Modeler thus serves as a common VPL for a variety of open source GIS software. In the right part of the input symbol and the symbol that represents the operation, the icons for deletion (cross) and editing (pencil or dots in v 3.x) can be found. The output data rectangle

does not have these two icons, because deleting or changing the operation automatically deletes the blue symbol of output data. In version 3.x, an icon allows setting output data as temporary.

The connectors are curved lines in all versions of the Processing Modeler. Connectors start and end at points marked Out / In.

The "plus" icon above operation symbols is used to expand the list of operation parameters. However, the parameter values are not displayed, only their names. A simple list of parameter names is not very beneficial to users; the actual parameter values would be more useful.

Contrary to previously presented visual syntax (ModelBuilder, Macro Modeler, Spatial Model Editor), operations are not connected by intermediate data symbols, but white operation rectangles are connected directly. It is possible to embed an existing model as sub-model. Python scripts can also be included in the model. The temporary disabling of operations in a model is possible (as in the AutoCAD workflow diagram).

The model is saved in a structured text file with the extension * .model.

The important feature of Processing Modeler (contrary to ModelBuilder and others) is that a model is not designed for existing concrete data. The inputs are supposed to be parametrical – users are required to input data to run of the model. The Processing Modeler in QGIS v. 3.x offers more input data types and more geoalgorithms than older versions 2.x. There are visible improvements in the functionality of Processing Modeler.

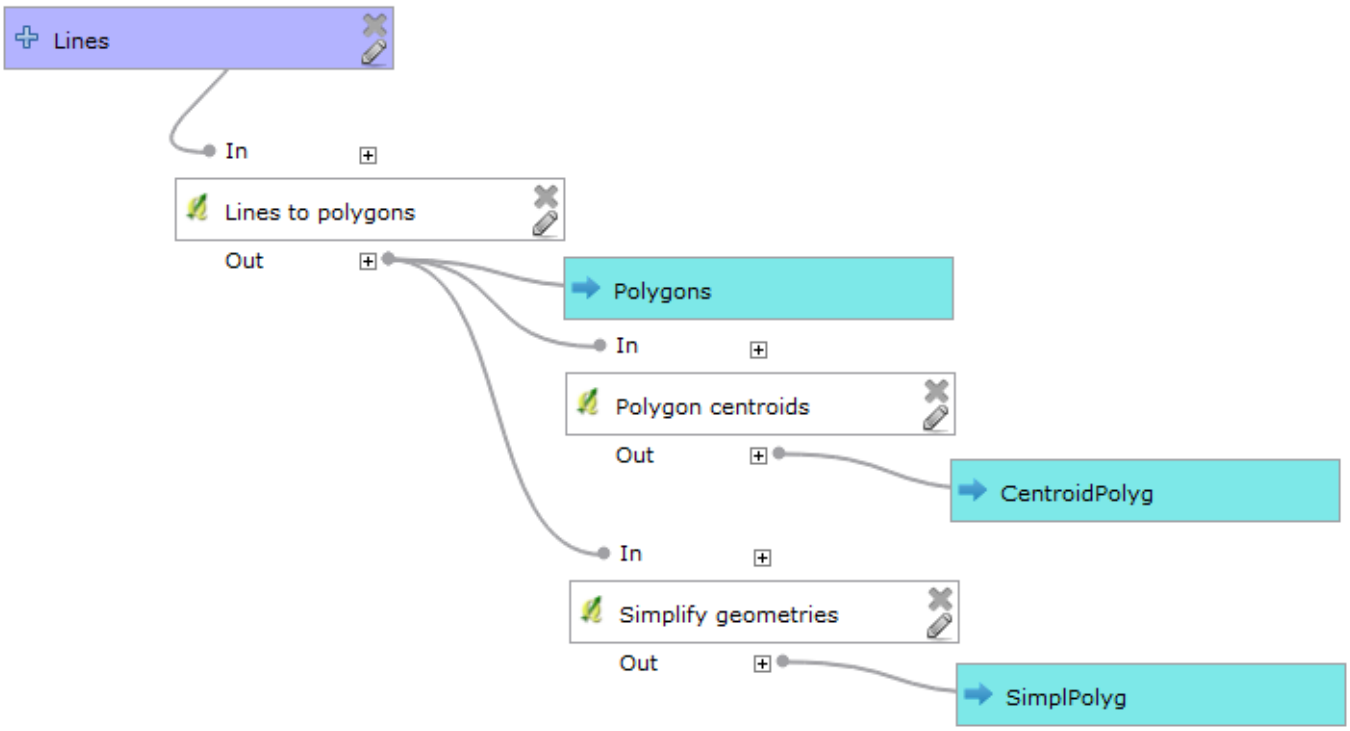


Figure 8. Model in Processing Modeler in QGIS, version 2.x. Source: author.

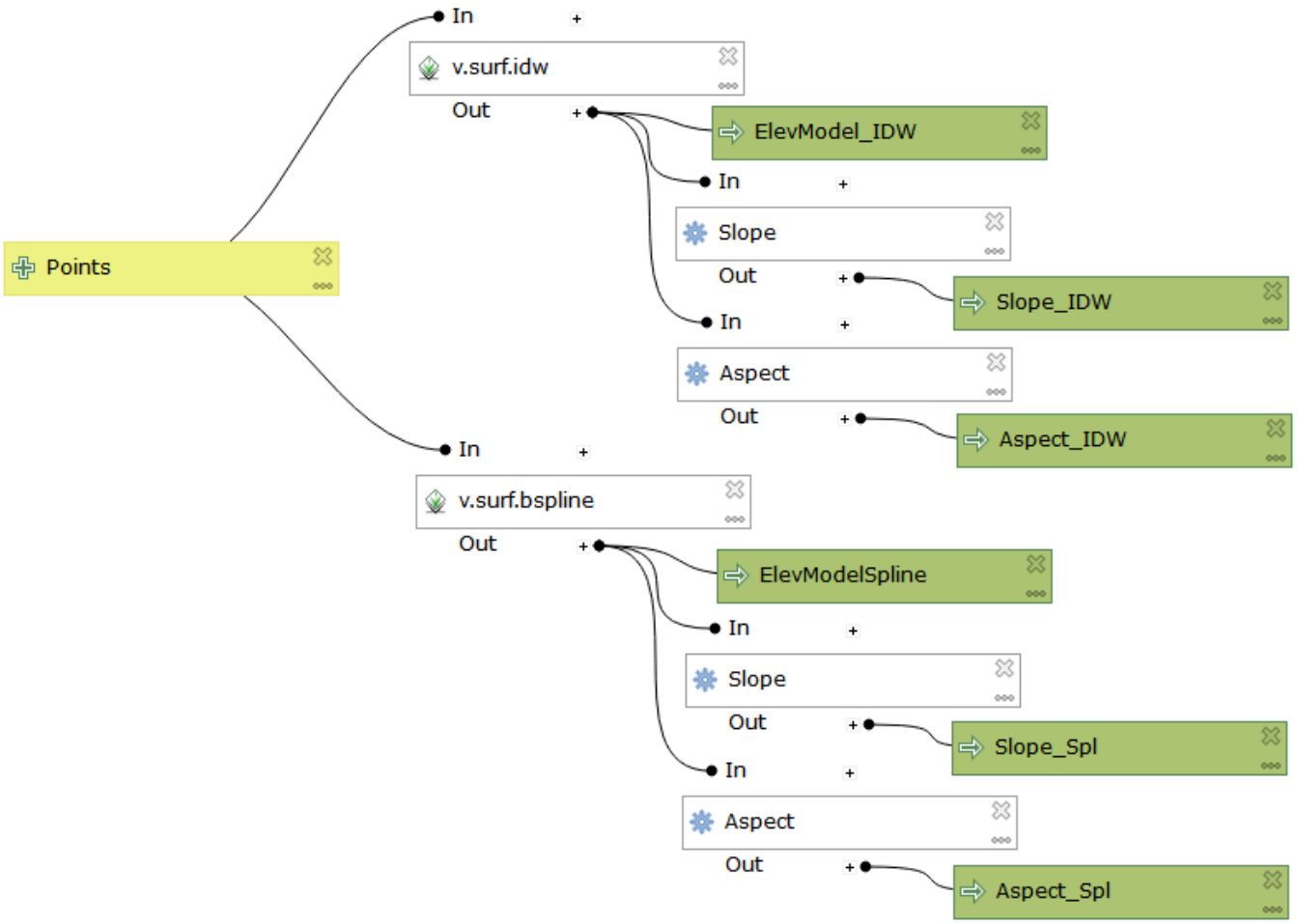


Figure 9. Example of a model from QGIS's Processing Modeler, version 3. Source: author.

4.7 Graphical Modeler in GRASS GIS

GRASS GIS has a Graphical Modeler (Landa n.d.). The graphical vocabulary was inspired by ArcGIS ModelBuilder. An oval symbol is used for data (vector - pink, raster - purple, 3D raster - green oval). Data can be set as temporary, i.e. data between operations, and will not be permanently stored. Temporary data will have a dashed border. The operation symbol is a green rectangle. In addition, it is possible to define loops (yellow rectangles with rounded corners) and If-Then-Else branching (white diamond). Comments can be written within operation rectangles or as separate rectangular symbols with orange fill and dashed outline. Operations have a number in parentheses before the operation name, which indicates the order of execution. Comments are also numbered (Figure 10).

The model can be parameterized, which means that input data, operation arguments and output data names can be entered at the beginning of the execution. This is denoted by a bold outline around the operation rectangle and the corresponding ovals of input/output data. This can be observed in the first operation, (1) v.surf.rst, in Figure 10. Operations may be temporarily disabled in the model: their color fill changes to gray, and the border to a dashed line. Connectors are straight black lines with end arrows. Connectors automatically flip 90° if necessary, and for the representation of loops. The symbols in the model need to be aligned manually, as automatic alignment of symbols is not available.

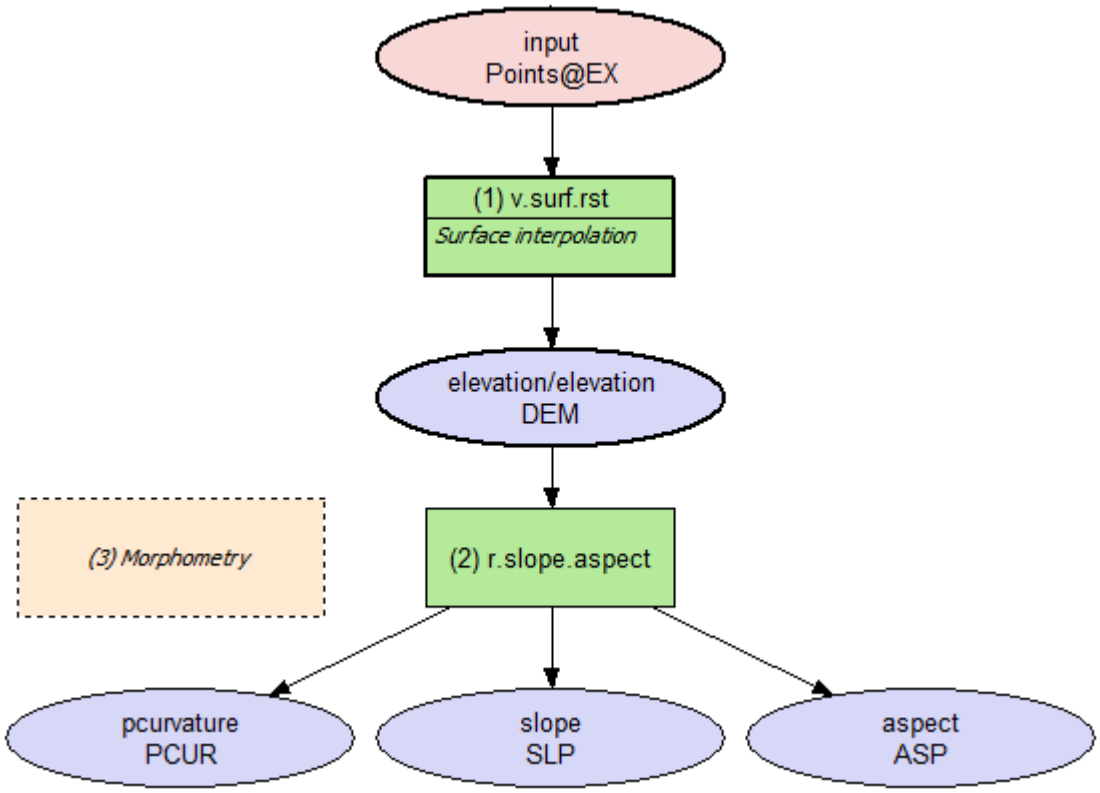


Figure 10. GRASS GIS Model in Graphical Modeler. Source: author.

4.8 Other VPLs for GIS Applications

Other VPLs in GIS exist. But the author of this article has no practical experience with them to fully describe them and document the graphical vocabulary, syntax and functionality in this text. An interesting one is **Rhino3D Grasshopper 3D**. It is very popular and extensively used in design fields. It is also quite mature, as it was launched in 2007. Grasshopper uses plugins like **Docofossor** for terrain modelling, **Bison** for terrain mesh, and **Groundhog** for terrain, flow and plants modelling. Hurkxkens and Bernhard (2019) provide a good example of using Docofossor.

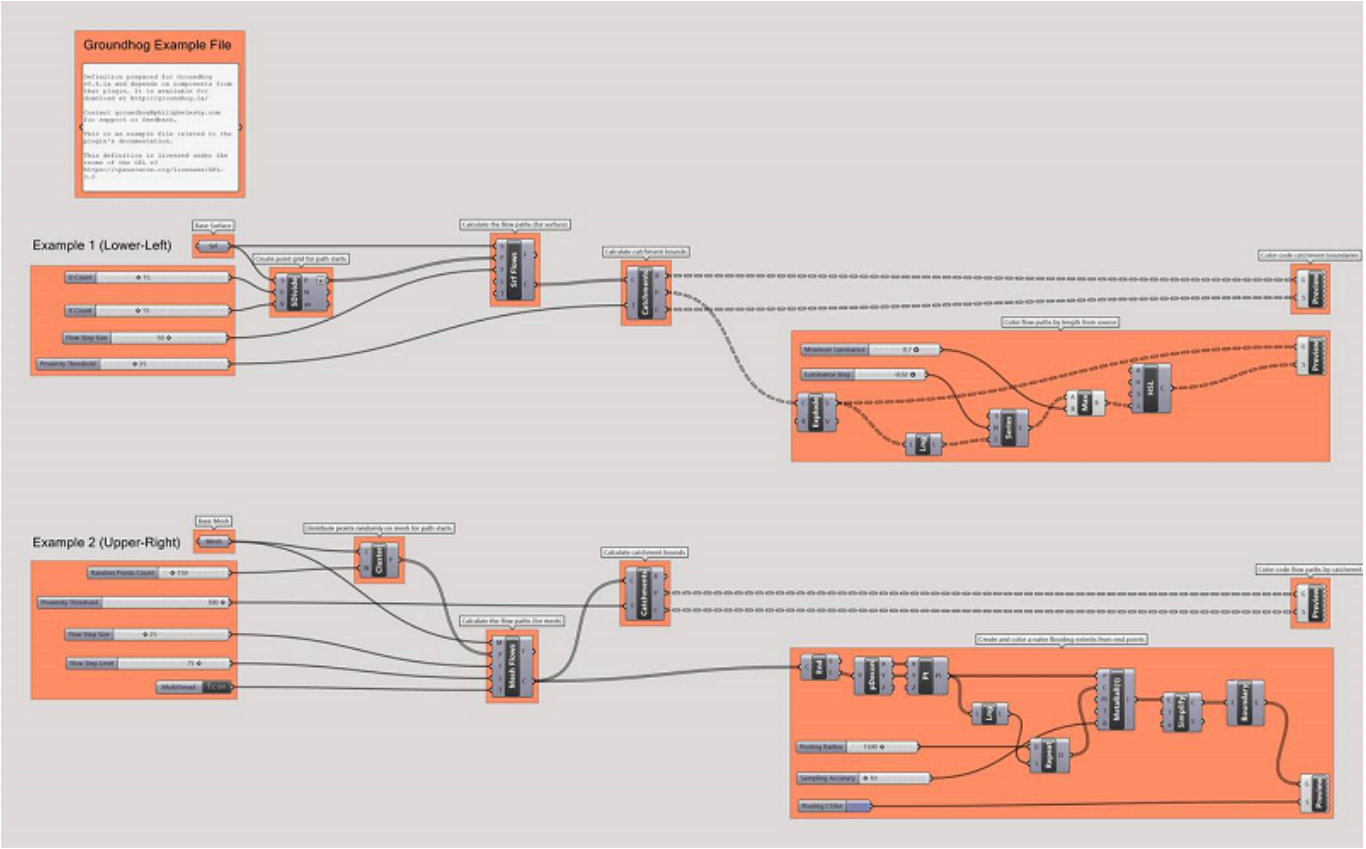


Figure 11. Grasshopper documentation demonstrating how to use the flow and catchment analysis for Surface and Mesh form. Source: Groundhog.

Firm Safe Software makes the FME (Feature Manipulation Engine) software. FME is a comprehensive software package designed to help manage a wide variety of spatial databases and convert them to a file format that facilitates processing in other third-party tools. For example, ArcGIS offers FME and its graphical editor under the Data Interoperability extension. FME contains over 450+ data converters. Among the formats that could be converted are databases like PostgreSQL, Geodatabase, ArcSDE, Oracle, JSON, SQL, Shapefile as well as non-native GIS applications, like MapInfo or Smallworld, Excel spreadsheets, CAD designs, raster images, LiDAR, and Autodesk dataset formats. The details of conversions are designed graphically by a VPL where curved lines connect the source and target data sets and, in detail, corresponding attributes for transformation (Figure 12).

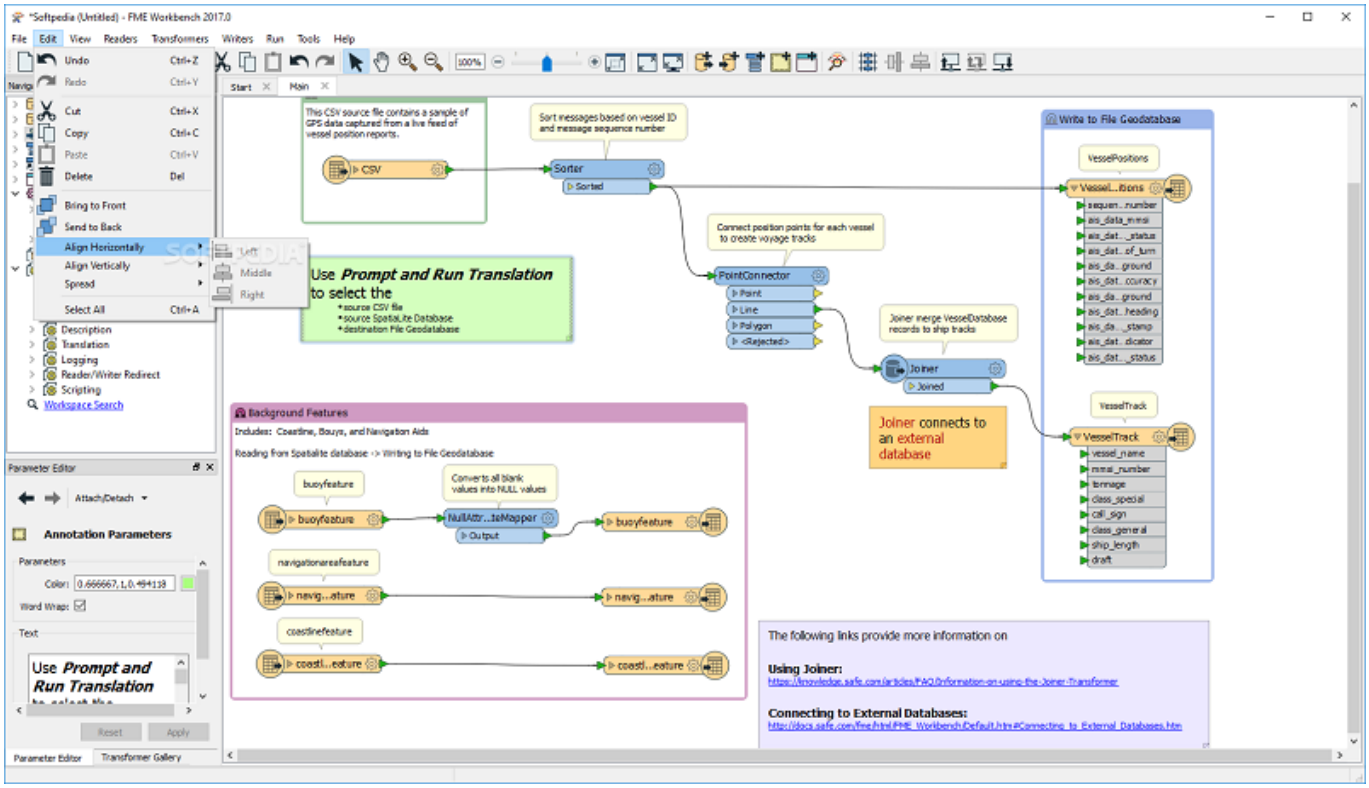


Figure 12. Visual design of transformation by FME. Source: Softpedia.

5. Evaluation of Effective Cognition

To compare and evaluate VPLs we can list functionalities or compare the appearances of graphical vocabulary. Functionalities very often depend on the number of spatial geoprocessing functions and image processing functions implemented in a concrete GIS application.

From the present overview of VPLs in GIS applications, it is evident that they differ in graphical vocabulary. In addition, the syntax (rules for connecting elements) and semantics differ as well. Besides appearance, the evaluation could be concentrated on the number of functionalities in a graphical editor, like zoom in/out of the diagram, disabling selected functions etc. Some VPLs have good functionalities, like automatic alignment, custom labels or elements sizes (Dobesova 2013a; Dobesova and Dobes 2014). These functions make the VPLs more usable for users, especially for novices in programming (Dobesova 2012).

Moody's "Physics of Notations" method can be used for a systematic evaluation of VPLs (Moody 2009a; Moody 2009b; Moody 2010) . It is suitable for any diagramming and therefore, for VPLs in GIS applications. The goal of the evaluation is to collect information regarding how a graphical notation promotes "Effective Cognition." The "Physics of Notations" method evaluates graphical notations through nine principles. For instance, the principle of "Perceptual Discriminability" recommends using different shapes and colors for symbols, as in ModelBuilder where orange for iterator, yellow for process and blue/green oval for data are used (Figure 13) (Dobesova 2013a).



Figure 13. Example of symbols in ModelBuilder to show differences in shapes and colors. Source: author.

The Spatial Model Editor in ERDAS IMAGINE scores very high on the principles of “Semiotic Clarity“ and ”Semantic Transparency.“ The big inner icons help to distinguish between symbols and help to intuit their meanings (Figure 14).

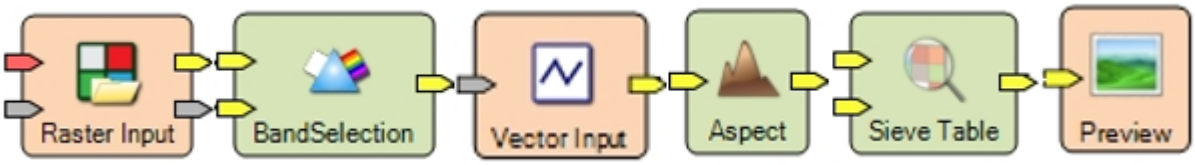


Figure 14. Example of some basic graphical elements in the Spatial Model Editor with high semantic transparency. Source: author.

An important function is the ability to nest sub-models. Also important is the ability to divide a very large diagram into groups, like what modularization in AutoCAD Map 3D Workflow Designer (Sequence activity and Parallel activity) or grouping in ArcGIS Pro ModelBuilder do. These features follow the principle of “Manageable Complexity.“

The scientific group at the Palacky University, Olomouc, Czech Republic (Department of Geoinformatics) conducted a systematic evaluation of the “Effective Cognition“ of six VPLs in GIS according to the Physics of Notations. In addition, the perception and cognition of different workflow diagrams (models) were evaluated through gaze measurements using an eye-tracker. Some results are presented in articles about measurement in the eye-tracking laboratories (Dobesova 2016; Dobesova 2017). The comparison of the influence of the shape of connectors on diagram readability is particularly interesting, as well as the influence of colors on the discrimination of symbols. Reading patterns and diagram orientation also influence cognition (Dobesova 2016). In Figure 15 below, the thickness of violet lines expresses the number of readers’ gazes. Some readers skip between lines of flow, and there is evident under-reading of the right side of the diagram. This distortion of reading is also typical with lines of text.

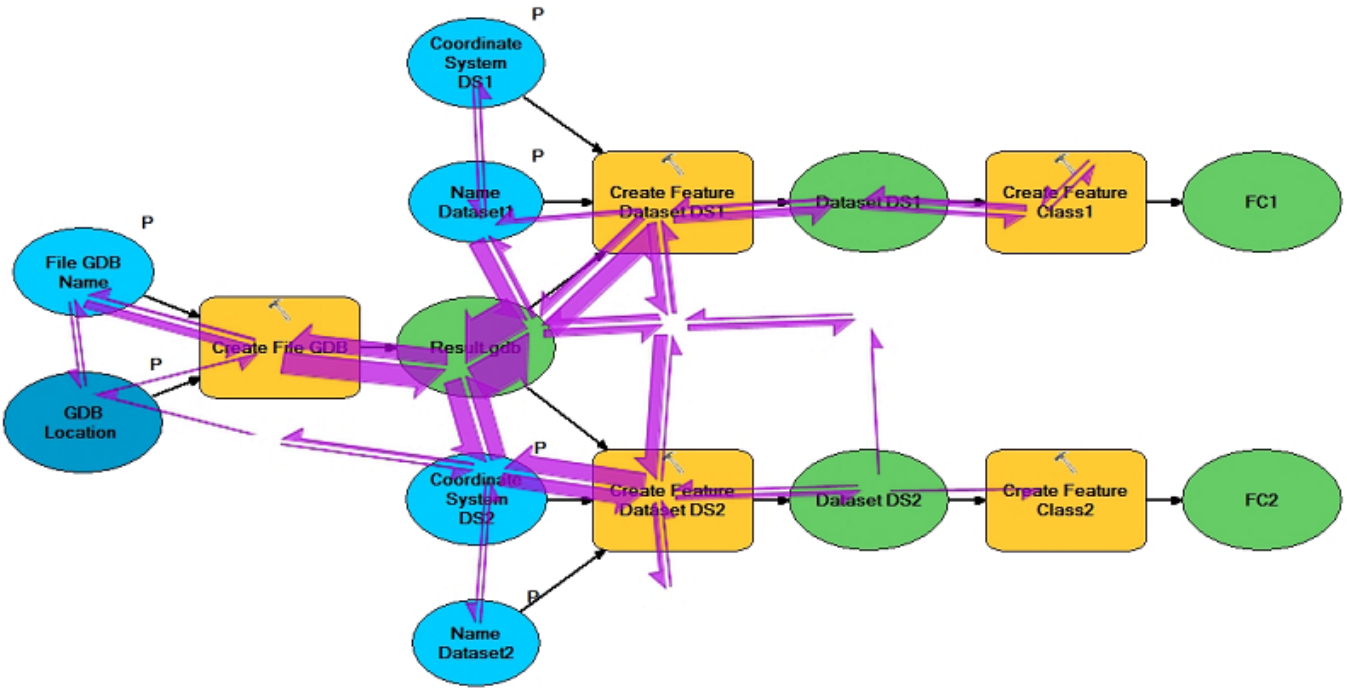


Figure 15. Aggregation of readers' gazes where line thickness expresses the number of gazes. Source: author.

The final recommendation for future users of VPLs in GIS: "Do not hesitate to utilize visual programming languages to automate simple everyday tasks."

References

Dobesova Z. (2017). Empirical Testing of Bends in Workflow Diagrams by Eye-Tracking Method. In: Silhavy R., Silhavy P., Prokopova Z., Senkerik R., Kominkova Oplatkova Z. (eds) Software Engineering Trends and Techniques in Intelligent Systems. CSOC 2017. Advances in Intelligent Systems and Computing, vol 575. Springer, Cham.

Dobesova, Z. (2012). Visual programming for novice programmers in geoinformatics. In Proceedings of 12th International Multidisciplinary Scientific GeoConference, Albena, Bulgaria, June 17-23, 2012; pp. 433- 440.

Dobesova, Z. (2013a). Strengths and weaknesses in data flow diagrams in GIS. In Proceedings of Computer Sciences and Applications (CSA), 2013 International Conference Wuhan, China; pp. 803-807.

Dobesova, Z. (2013b). Using the Physics of Notations to analyse ModelBuilder diagrams. In Proceeding of 13th International Multidisciplinary Scientific GeoConference, Albena, Bulgaria; pp. 595-602.

Dobesova, Z. (2014). Data flow diagrams in geographic information systems: a survey. In Proceedings of 14th SGEM GeoConference on Informatics, Geoinformatics and Remote Sensing, Albena, Bulgaria, June 19-25, 2014; pp. 705-712.

Dobesova, Z. (2016). Student reading strategies of GIS workflow diagrams. Journal of

Advances in Social Science, Education and Humanities Research, 70: 319-325.

Dobesova, Z., & Dobes, P. (2014). Differences in Visual Programming for GIS. Applied Mechanics and Materials, 519–520, 355–358.

ERDAS IMAGINE (1993). PE&RS Journal. Bethesda, MD: American Society for Photogrammetry and Remote Sensing (ASPRS), May, p 568.

Hexagon Geospatial. (2015). Automating Remote Sensing Workflows with Spatial Modeler.

Hurkxkens, I. and Bernhard, M. (2019). Computational Terrain Modeling with Distance Functions for Large Scale Landscape Design. Journal of Digital Landscape Architecture.

Landa, M. (n.d.). wxGUI Graphical Modeler. Accessed on February 20, 2020.

Moody, D. L. (2009a). Theory development in visual language research: Beyond the cognitive dimensions of notations. In Proceedings of 2009 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), Corvallis, OR, USA; pp. 151-154

Moody, D. L. (2009b). The "Physics" of Notations: Toward a Scientific Basis for Constructing Visual Notations in Software Engineering. IEEE Transactions on Software Engineering, 35, 756-779.