Proposal of a Python interface to OpenMI, as the base for open source hydrological framework

Robert Szczepanek
Division of Hydrology, Cracow University of Technology
Poland
Email: robert szczepanek.pl

Keywords: hydrological framework, Python, OpenMI, open source, Water Framework Directive

Abstract

Hydrologists need simple, yet powerful, open source framework for developing and testing mathematical models. Such framework should ensure long-term interoperability and high scalability. This can be done by implementation of the existing, already tested standards. At the moment two interesting options exist: Open Modelling Interface (OpenMI) and Object Modeling System (OMS). OpenMI was developed within the Fifth European Framework Programme for integrated watershed management, described in the Water Framework Directive. OpenMI interfaces are available for the C# and Java programming languages. OpenMI Association is now in the process of agreement with Open Geospatial Consortium (OGC), so the spatial standards existing in OpenMI 2.0 should be better implemented in the future. The OMS project is pure Java, object-oriented modeling framework coordinated by the U.S. Department of Agriculture. Big advantage of OMS compared to OpenMI is its simplicity of implementation. On the other hand, OpenMI seems to be more powerful and better suited for hydrological models. Finally, OpenMI model was selected as the base interface for the proposed open source hydrological framework.

The existing hydrological libraries and models focus usually on just one GIS package (HydroFOSS – GRASS) or one operating system (HydroDesktop – Microsoft Windows). The new hydrological framework should break those limitations. To make hydrological models' implementation as easy as possible, the framework should be based on a simple, high-level computer language. Low and mid-level languages, like Java (SEXTANTE) or C (GRASS, SAGA) were excluded, as too complicated for regular hydrologist. From popular, high-level languages, Python seems to be a good choice. Leading GIS desktop applications – GRASS and QGIS – use Python as second native language, providing well documented API. This way, a Python-based hydrological library could be easily integrated with any GIS package supporting this programming language. As the OpenMI 2.0 standard supported interfaces only for Java and C#, the Python interface for OpenMI standard, presented in this paper, is the first step done towards the open and interoperable hydrological framework. GIS-related issues of the OpenMI 2.0 standard are also outlined and discussed.

Introduction

Mathematical modelling in hydrological sciences gets use of geospatial functions from early 1990s [15]. Recent development in remote sensing and automatic data acquisition technologies lead to increase of available data. Analysis and processing of those data in distributed models

is very difficult without access to geospatial systems. HydroGIS conferences held in 1993 and 1996 in Vienna have shown the big interest in practical application of geographical information systems (GIS) in the field of hydrology. At the Cracow University of Technology (Poland), in late 1980s, we have developed a GIS based, distributed hydrological model WISTOO [27]. The model was successfully implemented in several Polish and few European catchments. Unfortunately, due to the closed license and monolithic structure in C language, the model is hard to maintain and develop. We faced a problem of how to build a new modern environment for hydrological modelling.

In modular programming, software is composed of separate, interchangeable components called **modules**, accomplishing one or more functions. The modules improve development of software by defining logical boundaries between the program components. This makes modular system more reusable than monolithic one, as modules can be reused in other programs. Example of such module is r.watershed in GRASS [18].

Modules are collected into **programs** (packages, toolsets) or **libraries**. GRASS and QGIS [29] are examples of programs, and their goal is to accomplish certain functions. The libraries contain a code that provide services to independent programs. The GDAL [9] and PROJ4 projects are examples of libraries.

The modules are typically incorporated into the program through **interfaces**. The interfaces are abstract types exposing internal behaviours to external modules.

A **framework** can be seen as a collection of software libraries with defined application programming interfaces (API). It is a foundation structure for developing applications. In terms of software design, the framework is a reusable software template, or skeleton, from which key enabling and supporting services can be selected, configured and integrated with the application code [21]. In general, an open source hydrological framework composes of common interfaces with underlying libraries, and its main goal is to meet the needs of hydrological modellers, by easy customization and adaptation of available libraries (Fig. 1).

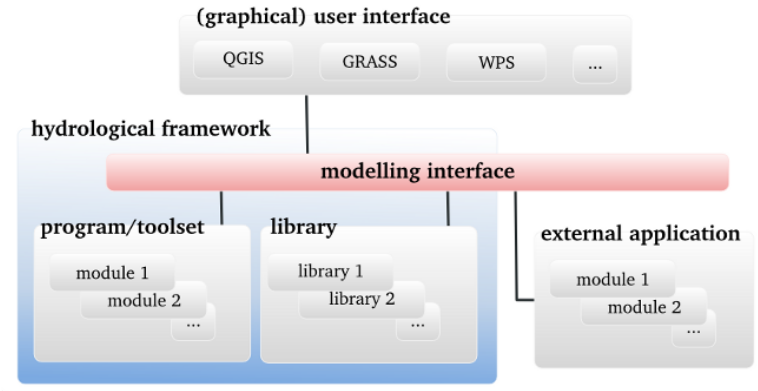

Figure 1: General structure of open source hydrological framework

There are two general approaches to linking hydrology and GIS. First one is based on the development of hydrological functions within the GIS environment. Such models are easier to integrate with GIS and provide better interoperability. Examples of this approach can be found in GRASS [18] or SAGA [12], module-based systems. An alternative approach focuses on hydrological functions, and GIS capabilities are treated as additional. In such a case, it

is very hard and time consuming to build geospatial capabilities of the system from scratch. It is much easier to build a hydrological system on the top of the existing applications (like QGIS) or libraries (like GDAL), than adding geospatial functionalities to the hydrological system. In that sense, proper selection of GIS foundation is the crucial step. Hopefully, there is a big diversity of GIS projects in the world of Free and Open Source Software for Geomatics (FOSS4G). On the other hand, the hydrological framework should not be coupled with just one geospatial system.

There are many great sources of code for such base GIS system: GRASS, SAGA, QGIS, Thurban, Whitebox GAT or HidroSIG [28]. There exist also ongoing projects to integrate some of the mentioned applications. Several programs are already able to use the SEXTANTE [22] library (see uDig [36], gvSIG [34], OpenJUMP [36]) or GRASS modules (see QGIS). An interesting project – QGIS Processing Framework – was also run to build an environment for execution of modules from external projects (SAGA, GRASS, OTB [13], OSSIM [36]) within the QGIS application.

There is a big variety of applications and libraries, but the problem I faced during research was how to provide a hydrologist-friendly programming environment. Once developed, the modules should be reusable and easy to modify, not only by the author, but also by other scientists. In many existing hydrological applications, based on the low-level computer languages, changes in the existing code for regular hydrologist are difficult. Instead of fighting with the code, one should focus on the problem to be solved. To reach this goal, a framework, easy to use and implement, and based on a high-level computer language is needed. Such framework should be available with standard interfaces, good documentation, tutorials and case studies.

Another important issue related to many existing applications is the lack of interoperability and redundancy. The redundancy of functions/modules can cause inconsistency, as the implementation methods can be different in different applications. It will be probably easier for hydrological community to develop and maintain just one implementation instance. In fact, there are already many elements of this puzzle, but they are hard to merge. To make hydrologist researcher life easier, it is not another hydrological application that is needed, but a library with interoperable functions of the basic hydrological processes. Based on a module from such library, any application can be build on top, and several alternative hypothesis or approaches can be tested easily. It should be also possible to use other external models compatible with the selected interface, or even to use a hydrological library as the Web Processing Service (WPS) back-end.

Modelling frameworks

In the year 2000, the European Union approved the European Water Framework Directive (WFD 2000/60/EC). WFD become an important document for the integrated river basin management and interdisciplinary studies not only on our continent. Many countries faced the problem of weak interoperability of the existing information systems, coming from different environmental domains. Additionally, trans-boundary issues and international cooperation become an important element of WFD reporting. Big diversity of standards in Europe is a fact; one of potential options was common versatile interface built on top of the existing systems. Based on the experience from previous projects, two groups initiated development

of completely different modelling frameworks: Object Modeling System (OMS) [7] and Open Modelling Interface (OpenMI) [26]. The assumptions of those two systems were different, so finally two different solutions were elaborated. The first one (OMS) was very simple and pragmatic, while the second one (OpenMI) was more sophisticated and strictly oriented to the WFD needs. The simpler framework allows only linear workflows of modules/models, while the second one enables feedbacks and looping.

OMS

Object Modeling System (OMS) framework has been developed in a joint approach by the U.S. Geological Survey (USGS), the U.S. Department for Agriculture (USDA) and the Friedrich-Schiller-University from Jena, Germany [7]. The prototype version OMS 1.0 was published in 2001; the latest version OMS 3.0, called Next Generation Modeling Framework (NGMF), has been released recently.

OMS is pure Java, lightweight, object-oriented modelling framework working in the NetBeans environment. OMS 3.0 provides programming interfaces in Fortran, C and C++. It supports geospatial integration, calibration tools and sensitivity analysis. In the newest version, a minimal invasive approach was applied, and, as a result, no framework data types and no interfaces were provided. OMS 3.0 is multithreaded. Its components are Plain Java Objects enriched with descriptive metadata by means of language annotations [8]. The annotations are being utilized to specify resources in a class, that relate to its use as a component. They allow for the extension of the Java programs with meta information that can be picked up from sources, classes, or at runtime.

The OMS component metadata implements just three methods – Initialize, Execute and Finalize. The components are designed with a standard well-defined interface in mind. They are self-contained units from the conceptual and technical perspective, and can be developed and tested individually [19].

Several environmental models are implemented using OMS 3.0. One of them is the Precipitation Runoff Modeling System (PRMS/OMS). One of the newest OMS implementations, interesting from hydrological point of view, is the jgrasstools project [14] based on the JGrass application. The jgrasstools is fast growing and very well documented library of basic hydrological and geomorphological algorithms.

OpenMI

OpenMI was developed in 2001 within the Fifth European Framework Programme as a part of the HarmonIT project, and later the OpenMI-LIFE projects. Partners of the project were: Natural Environment Research Council (UK), DHI (DK), Deltares (NL), Wallingford Software (UK), National Technical University of Athens (EL), University of Thessaly (EL), Aquafin (BE), VMM – AK (BE), Flanders Hydraulics Research (BE) and Université de Liège (BE). The idea was to easily combine programs from different providers, enabling a modeller to make free choice of the best model suited to the particular needs. The OpenMI standard is an interface definition for the computational core of the computer models in the water domain [25]. Model components that comply with this standard can, without any programming,

exchange data at run-time [10]. OpenMI is a pull-based architecture that consists of linked components (source components and target components) which exchange memory-based data in the single-threaded architecture.

The OpenMI standard is defined by a set of software interfaces that a compliant model or component must implement. The interfaces are available in the C# and Java languages. Version 2.0 of OpenMI released in November 2010 specifies the base interfaces and the extension supporting the time- and space-dependent component (OpenMI.Standard2.TimeSpace) [26].

To support the development of OpenMI compliant components, a Software Development Kit (SDK) has been provided for .NET developers [26]:

- Backbone – a default implementation for the majority of the OpenMI.Standard2 interfaces,
- DevelopmentSupport – some very generic support utilities,
- Buffer – utilities for timestep buffering and time interpolation and extrapolation,
- Spatial – utilities for spatial interpolation,
- ModelWrapper – utilities to facilitate wrapping of existing models.

The usage of the OpenMI namespace is the mandatory part of any OpenMI compliant software component. In standard documentation [26][24], list of interfaces is described, as minimal and as complete as possible, to define exactly the data that is being exchanged. Every compliant component must have an associated registration file in the XML format and implement the IBaseLinkableComponent interface.

The exchange items are used in the initial phase to define what the component can provide (as output), and what information the component accepts (as input). In OpenMI 2.0, values can be also transformed as needed with help of adopted outputs. The OpenMI does not use standardized data dictionaries, except SI units. The component life is divided into five phases: initialization, configuration, preparation, execution and completion. Every phase is well described in documentation [24].

In August 2011, the OpenMI Association and Open Geospatial Consortium (OGC) have signed the memorandum of understanding to cooperate in standards development and promotion of open standards related to computer modelling. At the moment, spatial operations are based on vector objects and no direct raster support is available (Fig. 2). Spatial elements are represented as points, lines, polylines, polygons or polyhedrons in element sets containing information about the georeference system in the form of WKT [20]. It is possible to link 1D and 2D models/components and exchange data between them in run time. To perform more advanced spatial operations, OpenMI authors prepared SDK (extension Spatial) and sample code files. From the beginning, OpenMI was released under the Lesser General Public Licence (LGPL). OpenMI 1.4 compliant software includes, i.a., InfoWorks, ISIS, SWAT, Sobek and MIKE.

Open source hydrological libraries and applications

The first group of interest are complete hydrological open source models. They are standalone applications or the applications created within one of the popular GIS platforms. The

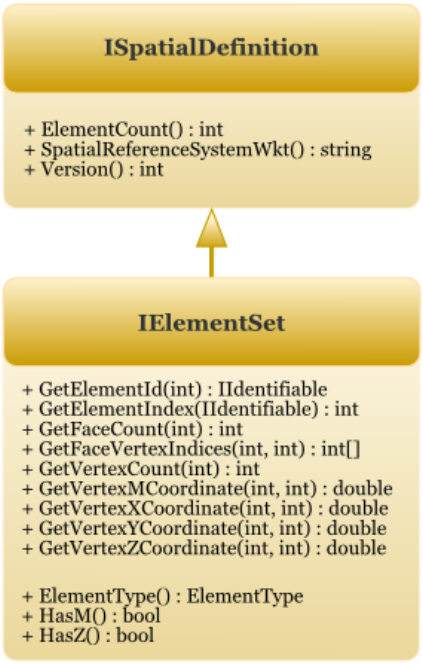

Figure 2: Interfaces related to spatial data in OpenMI 2.0

most popular and known open source hydrological models are:

- HydroFOSS [5], GIPE, TOPMODEL [4], ANSWERS [30] (built on the GRASS project),
- GEOtop [32] (JGrass),
- IHACRES [6] (SAGA),
- PIHM [37] (QGIS),
- HIS Desktop [1],
- HidroSIG [28],
- HydroPlatform [11] (Thurban),
- Kalypso [2],
- HYPE [16].

Of the mentioned above, the Kalypso model has been recently one of the most popular. There are also open source hydrological modules available in the proprietary software, like TauDEM [35] in ArcGIS.

Functions for geomorphological analysis based on the Digital Elevation Models can be found in almost every GIS package. Also, almost every GIS package can be used for hydrological pre- and post-processing of spatial data. There are however packages which contain much more advanced and oriented strictly to hydrological analysis functionalities such as:

- r.watershed, r.drain.r.flow, r.stream.* (GRASS),

- Terrain Analysis – Hydrology, Channels (SAGA),
- SEXTANTE,
- jgrasstools.

Many GIS applications have built-in functions used for hydrological purposes. There are also external spatial data analysis libraries like SEXTANTE [22][34]. This Java library contains more than three hundred algorithms for both raster and vector processing, and has bindings to GIS applications like gvSIG, uDig, OpenJUMP, Kosmo and ArcGIS [23]. There are also ongoing implementations of SEXTANTE as the back-end for GeoServer and 52North WPS Server [23]. This library includes 10 algorithms for the basic hydrological analysis, 15 for indices and other hydrological parameters, and 11 algorithms for the geomorphometry and terrain analysis.

Some packages (e.g. GRASS and SEXTANTE) have graphical modellers (user interfaces) for more user-friendly workflow design and implementation. This is substitution of the previously used scripting languages, for less experienced users. The mentioned graphical modellers are able to model only onedirectional linear workflows.

Having so many options to choose, proper selection of a package is the major problem. Direct comparison is not so easy. Open source packages give access to source code, but sometimes it is hard to analyse implementation of algorithms in different languages written by different programmers. So, is the implementation of the same algorithms comparable? In the last two years a synergy effect in the desktop GIS applications and libraries can be observed. Several FOSS4G projects cross-reference each other. The leader in this field is probably the QGIS project with a QGIS Processing Framework, aiming at development of generic framework for external packages integration. At the moment, the QGIS project has a part of GRASS modules included in the last releases, the SAGA modules are almost ready to work under QGIS, and the Orfeo Toolbox and OSSIM modules are planned as the next QGIS toolboxes.

An advantage of a library compared to an application is the fact that the library must be interoperable to survive, so its implementation should be simpler. A well-designed library should be interoperable not only in terms of the interface (hardcoding vs. OpenMI), but also in terms of the access method (direct access vs. WPS).

The presented libraries and applications represent only small part of all available resources.

Towards interoperable hydrological framework

The main goal of development of the hydrological framework is to provide a relatively easy but powerful environment for research in hydrological sciences. This is more oriented to educational purposes than operational hydrology.

Within the presented work, two steps towards developing the open source hydrological framework have been made. The first one was the analysis of potential elements to be used. It included GIS platforms, hydrological libraries and applications, modelling frameworks and interfaces. Of course, all of them are open source. Based on this analysis, decision on optimal platform selection was made. As result of this analysis, the second step was concentrated mainly on adoption of a selected interface standard for this project's purposes.

In general, when building an open source hydrological framework, the following goals were taken into consideration:

- free and open source license,
- implementation of one of the popular modelling frameworks; powerful, with long-time support,
- reusability of components,
- simplicity of coding and use of cross-platform computer language,
- use of already existing code and algorithms,
- scalable architecture,
- simplicity of implementation (research), but not computational efficiency (operational purposes),
- compatibility with popular GIS environments (both server and desktop),
- GUI independency.

Selection

Both presented frameworks (OpenMI and OMS) are modern, well designed and versatile. OpenMI has strong support from the leading water resources firms in Europe, while OMS is more U.S. public administration related. OpenMI pays much attention recently on the spatial aspects of modelling. OMS is Java-based while OpenMI implements both C# and Java interfaces. Big advantage of OMS compared to OpenMI is simplicity of implementation. One of the arguments in favour of OpenMI is that many of existing hydrological models, listed by the Community Surface Dynamics Modeling System, plan to implement this standard. There are already examples of OpenMI use as a link between different models, like PIHM and TauDEM [17], HIS Desktop [1], but mainly on Windows platform. Finally, the OpenMI standard has been selected as a base for the developed hydrological framework.

Through the OpenMI interface, it will be possible to access models from different consumers. In the first stage, the access from desktop applications, like QGIS and GRASS, will be tested. Web services like WPS, installed as server applications, are the second potential consumer. Finally, it will be possible to get use of any OpenMI compliant, external model (component). In the DRIHMS project [33], focused on high performance computing, OpenMI is used as the interface between components.

There are many valuable open source hydrological libraries and modules to be included in the developed hydrological framework. The first practical limitation is the problem of platform dependency. Many existing hydrological applications focus on one operation system only. HIS Desktop [1], developed by the Consortium of Universities for the Advancement of Hydrologic Sciences (CUAHSI), is good example of that. It uses C#/.NET limiting HIS Desktop usage to Microsoft Windows operating system. And even the Mono project, does not seem to be the best direction to solve this problem. That is why the cross-platform languages, like Java or Python, are probably a better choice.

The existing GIS applications with hydrological functions use very different computer languages. The most popular are:

- C/C++ – GRASS, QGIS, SAGA,
- Java – uDig, SEXTANTE,
- C# – HydroDesktop,
- Python – Thurban.

There are, however, bindings to another languages, for example, the GRASS, QGIS and SAGA plugins, can be easily created in Python. As the hydrological framework should be used mainly be non-professional programmers, the most popular but difficult low-level languages, like C and Java, were excluded [31]. A perfect language should be easy to learn and implement. From the world's TOP10 languages, only Python is a high-level, open source and cross-platfom language with good support by GIS applications. According to the Tiobe portal, Python was "Programming Language of the Year 2010" with the highest rise in the ratings. The Python interpreter is not the fastest engine, especially when compared with compiled languages like C++, but this can be optimized using tools like Theano [3] to compile mathematical expressions and get use of the Graphics Processing Unit. Only HydroPlatform uses Python as the native language. Two popular desktop GIS applications (QGIS and GRASS) use Python as the second language in development. Finally, Python have been selected as the basic language for hydrological framework.

The only problem is that there are no OpenMI interface standards for Python. Having OpenMI as the modelling framework and Python as the basic language, the important step in framework implementation was the translation of the C# OpenMI specification to Python.

Implementation

As there are substantial differences between the source (C#, Java) and target (Python) languages of the OpenMI interfaces, to keep compatibility with the source, no major changes have been done and all the comments were copied from the source files.

The first significant difference between the source and target language is related to the type definition. Python is dynamically typed and does not require an explicit declaration of variables before they are used. So the data types in a Python version of OpenMI have rather descriptive function and rely on the "duck typing" paradigm.

The second difference relates to the implementation of interfaces. The interface mechanism is typical for the C# and Java languages. In Python, the interfaces are often implicit and defined only by usage and documentation. Python support multiple inheritance, so I decided to implement the OpenMI interfaces as classes. This should give better picture of interfaces functionality to new developers.

The naming conventions and namespaces standards have not been changed yet and follow the OpenMI C# implementation.

As the hydrological framework should follow the open source philosophy, all code under development is available to the public in internet. The project was named Open Hydrology,

and most of its necessary infrastructure is hosted on the SourceForge portal (http://sourceforge.net/projects/openhydrology/).

Most of the OpenMI 2.0 specification standard was translated to the Python language. At this stage, no GUI and SDK for the Python version of OpenMI are available. Sample implementations, documentation, GUI for easy model access and workflow modeller will be built in the future.

Discussion

In the time when new projects are launched every day, more and more initiatives tend to integrate efforts to better use the available human resources. The GDAL project is a good example of this process. In the before-GDAL era, every GIS package used its own mechanism for data access. The same situation is now with hydrological models. There are several redundant models unable to exchange data with each other.

In order to provide good interoperability, the hydrological framework design is based on the mature and tested modelling framework OpenMI. There are some threats related to this choice. One of them is the complexity of OpenMI standard implementation.

Python as the basic language for new hydrological framework was relatively easy choice, as this computer language has already mature status in FOSS4G software. The problem is that in the hydrological domain it is not very popular language yet.

Translation of the OpenMI 2.0 standard to Python interfaces was the first step needed to start the development of the framework. There are two potential options for the second step. Development of new and complete OpenMI components in Python (Fig. 3a), as the long-term goal, is the first one. The second option, faster, but rather short-term, is the wrapping of the existing hydrological C modules (Fig. 3c), for example from GRASS. For this purpose, Python is a very good choice.

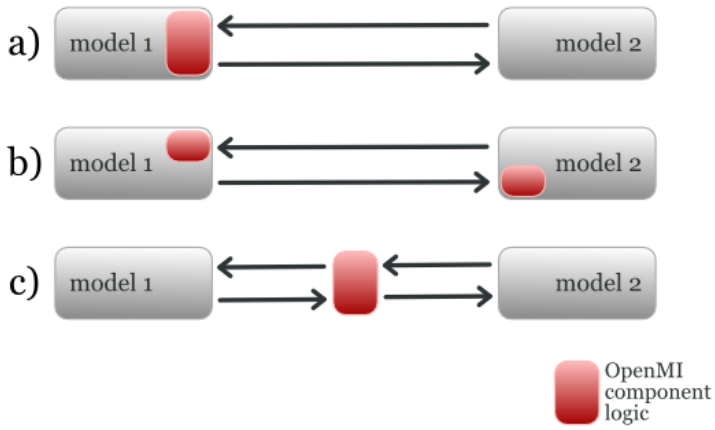

Figure 3: Alternative locations of OpenMI interfaces (component logic)

The OpenMI interfaces can be available at any level of the hydrological model. Preferably higher decomposition into elementary physical processes, not treating the whole model as one unit, will be better. This will give scalability and usability, but at the cost of higher initial

workload. I hope to attract the open source developers to the Open Hydrology project, as till now the OpenMI Association is concentrated mostly on the C# and commercial applications. OpenMI 2.0 is a new standard, and according information on the web page of the project, there are no compliant software available yet.

References

1. Ames, D. P., Horsburgh, J., Goodall, J., Whiteaker, T., Tarboton, D. and Maidment, D. (2009). Introducing the Open Source CUAHSI Hydrologic Information System Desktop Application (HIS Desktop), 18th World IMACS/MODSIM Congress, Cairns, Australia.

2. Belger, G., Haase, M., Jung, T. and Lippert, K., (2009). A GIS-based Platform for Environmental and Water Resources Modeling – Kalypso Open Source, GEO Informatics magazine.

3. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D. and Bengio, Y., (2010). Theano: A CPU and GPU Math Expression Compiler. Proceedings of the Python for Scientific Computing Conference (SciPy) 2010. Austin, TX

4. Beven, K.J., Lamb, R., Quinn, P., Romanowicz, R. and Freer, J. (1995). TOPMODEL, in Computer Models of Watershed Hydrology, Singh V.P. (Ed.), Water Resources Publications, 627-668.

5. Cannata , M., (2006). A GIS embedded approach for Free & Open Source Hydrological Modelling, PhD dissertation, Politecnico di Milano.

6. Croke, B.F.W., Andrews, F., Jakeman, A.J., Cuddy, S. and Luddy, A., (2005). Redesign of the IHACRES rainfall-runoff model, Proceedings of the 29th Hydrology and Water Resources Symposium, Engineers Australia

7. David, O. and Krause, P., (2002). Using the Object Modelling System for future proof of hydrological model development and application. Proceedings of the Second Federal Interagency Hydrologic Modeling Conference, Las Vegas, NV, July 28 – August 2, 621-626.

8. David, O., Ascough II, J., Leavesley, G., and Ahuja, L., (2010). Rethinking modeling framework design: Object Modeling System 3.0. IEMSS 2010 International Congress on Environmental Modeling and Software – Modeling for Environment's Sake, Fifth Biennial Meeting, July 5-8, 2010, Ottawa, Canada; Swayne, Yang, Voinov, Rizzoli, and Filatova (Eds.)

9. Donnelly, F.P., (2010). Evaluating open source GIS for libraries, Library Hi Tech, Vol. 28 Iss: 1, 131-151.

10. Gregersen, J.B., Gijsbers P.J.A. and Westen S.J.P. (2007). OpenMI: Open Modelling Interface. Journal of Hydroinformatics 9(3), 175-191.

11. Harou, J., Pinte, D., Hansen, K., Rosenberg, D., Tilmant, A., Medellin-Azuara, J., Pulido-Velazquez, M., Rheinheimer, D., Matrosov, E., Reynaud, A., Kock, B., and Vicuna, S., (2009). HydroPlatform.org – an open-source generic software interface and web

repository for water management models. International Symposium on Environmental Software Systems, ISESS 2009, Venice, Italy.

12. Hengl, T., Grohmann, C.H., Bivand, R.S., Conrad, O. and Lobo, A., (2009). SAGA vs GRASS: A Comparative Analysis of the Two Open Source Desktop GIS for the Automated Analysis of Elevation Data. Geomorphometry 2009 Conference Proceedings, In: Geomorphometry 2009, Edited by R. Purves, S. Gruber, R. Straumann and T. Hengl. University of Zurich, Zurich, 22-27.

13. Inglada, J. and Christophe, E., (2009). The Orfeo Toolbox remote sensing image processing software, Geosciences and Remote Sensin Symposium, 2009 IEEE International, IGARSS 2009, Cape Town, IV 733-736.

14. jgrasstools project, Available at: http://code.google.com/p/jgrasstools/

15. Kopp, S.M., (1996). Linking GIS and hydrological models: where we have been, where we are going? Proceedings og HydroGIS 96: Application of Geographic Information Systems in Hydrology and Water Resources. IAHS Publ. no.235. 133-139.

16. Lindström, G., Pers, C.P., Rosberg, R., Strömqvist, J. and Arheimer, B, (2010). Development and test of the HYPE (Hydrological Predictions for the Environment) model – A water quality model for different spatial scales. Hydrology Research 41.3-4:295-319.

17. Lu, B. and Piasecki, M., (2008). Development of an Integrated Hydrologic Modeling System for Rainfall-Runoff Simulation, American Geophysical Union, Fall Meeting 2008, abstract #H41G-0953,
Available at: http://adsabs.harvard.edu/abs/2008AGUFM.H41G0953L

18. Neteler, M., Bowman, M.H., Landa, M. and Metz, M., (2012). GRASS GIS: A multipurpose open source GIS. Environmental Modelling & Software.

19. Object Modeling System (OMS) project, (2011).
Available at: http://www.javaforge.com/project/oms

20. OGC, (2002). The OpenGIS Abstract Specification Topic 2: Spatial referencing by Coordinates OGC 01-063r2, OpenGIS Consortium Inc.

21. OGC, (2012). OGC Glossary, Available at: http://www.opengeospatial.org/ogc/glossary/

22. Olaya, V., (2012). SEXTANTE – Spatial Data Analysis Library,
Available at: http://www.sextantegis.com/

23. Olaya, V. and Gimenez, J.C., (2011). SEXTANTE, a versatile open–source library for spatial data analysis.

24. The OpenMI Association, (2010). OpenMI Standard 2 Reference for the OpenMI (Version 2.0). Part of the OpenMI Document Series.

25. The OpenMI Association, (2010). Scope for the OpenMI (version 2.0). Part of the OpenMI report series.

26. The OpenMI Association, (2010). OpenMI Standard 2 Specification for the OpenMI (Version 2.0). Part of the OpenMI Document Series.

27. Ozga-Zielińska, M., Gadek, W., Książyński, K., Nachlik, E. and Szczepanek R., (2002). Mathematical model of rainfall-runoff transformation – WISTOO, Mathematical Models of Large Watershed Hydrology, Ed. V.P.Singh, D.K.Frevert, Water Resources Publications, 811-860.

28. Poveda, G., Mesa, O. J. and Velez, J. I., (2007). HidroSIG: An interactive digital atlas of Colombia's hydro-climatology, Journal of Hydroinformatics, 9 (2), 145-156.

29. Quantum GIS Development Team, (2011). Quantum GIS Geographic Information System. Open Source Geospatial Foundation Project. Available at: http://qgis.osgeo.org

30. Rewerts, C.C. and Engel, B.A., (1993). ANSWERS on GRASS: Integration of a watershed simulation with a geographic information system. Abstracts, Proc., 8th Annu. GRASS GIS User's Conf. and Exhibition.

31. Rey, S.J., (2008). Show Me the Code: Spatial Analysis and Open Source, unpublished, Available at: http://geodacenter.asu.edu/2008_11
http://geodacenter.asu.edu/2008_11

32. Rigon, R., Bertoldi, G. and Over, T. M., (2006). GEOtop: A Distributed Hydrological Model with Coupled Water and Energy Budgets., Journal of Hydrometeorology, Vol. 7, No. 3, 371-388.

33. Schiffers, M., Kranzlmuller, D., Clematis, A., D'Agostino, D., Galizia, A., Quarati, A., Parodi, A., Morando, M., Rebora, N., Trasforini, E., Molini, L., Siccardi, F., Craig, G.C. and Tafferner, A., (2011). Towards a grid infrastructure for hydro-meteorological research, Computer Science, Vol. 12, 45-62.

34. Schröder, D., Hildahb, M. and David, F., (2010). Evaluation of gvSIG and SEXTANTE Tools for Hydrological Analysis. 6th International gvSIG Conference. Available at: http://jornadas.gvsig.org/6as-jornadas-gvsig/descargas/articles

35. Tarboton, D. G. and Baker, M.E., (2008). Towards an Algebra for Terrain-Based Flow Analysis, in Representing, Modeling and Visualizing the Natural Environment: Innovations in GIS 13, Edited by N. J. Mount, G. L. Harvey, P. Aplin and G. Priestnall, CRC Press, Florida.

36. Tsou, M.H. and Smith, J., (2011). Free and Open Source Software for GIS education, Department of Geography, San Diego State University, (white paper)

37. Yizhong, Q., (2004). An integrated hydrologic model for multi-process simulation using semi-discrete finite volume approach, PhD Thesis, The Pennsylvania State University. Available at: http://www.pihm.psu.edu/Downloads/Articles/qu_thesis.pdf