OpenFLUID : a software environment for modelling fluxes in landscapes

Fabre, J.-C. (1); Louchart, X. (1); Colin, F. (3); Dagès, C. (1); Moussa, R. (1); Rabotin, M. (1); Raclot, D. (2); Lagacherie, P. (1); Voltz, M. (1)

(1) INRA, UMR LISAH, 2 Place Viala, 34060 Montpellier, France, fabrejc@supagro.inra.fr

(2) IRD, UMR LISAH, 2 Place Viala, 34060 Montpellier, France

(3) SupAgro, UMR LISAH, 2 Place Viala, 34060 Montpellier, France

Abstract: Farmed landscape functioning and changes in time and space are mainly influenced by multiple bio-physical processes, human activities, socio-economic driving factors and by the interactions between all of them, resulting in a complex multi-scale system. Thus, modelling and simulation of such systems is very hard and requires both solid scientific background and the development of specific software tools.

The OpenFLUID platform was developed in this way, that provides a modern, flexible and easy-to-use modelling framework, allows and facilitates the implementation and/or development of single process components to be used, reused and coupled according to the modelling objectives, the spatial scale and the data. OpenFLUID is mainly based on a spatially distributed modelling approach, where elements and spatial patterns of the landscape are in interaction with the simulated processes (e.g. agro-bio-physical fluxes) and can influence or be influenced by each-other.

We describe here the main components of the OpenFLUID platform and their objectives, basic principles and technical features: the simulation engine (OpenFLUID-Engine), the graphical user interface (OpenFLUID-Builder) and the community web site (OpenFLUID-web). As an example, the implementation of the hydrological model MHYDAS is presented. The major OpenFLUID improvements and perspectives are also presented.

Keywords: Modelling Framework; Simulation; Distributed model; Landscape; Fluxes

Introduction

Farmed landscape functioning and changes in time and space are influenced by multiple bio-physical processes (e.g. hydrology, erosion), human activities (e.g. agricultural practices, land-use planning), socio-economic driving factors and policies (e.g. tourism activities, extensive agriculture) and by the interactions between all of them. These strong interactions and feed-backs result in a complex multi-scale system, for which an inter-disciplinary approach is needed. Thus, modelling and simulation (M&S) of such systems is very hard and must address both scientific, technical and organisational issues such as:

- if existing models present potentiality to simulate part of the landscape functioning, they need to be adapted to the context and objectives of the studies
- a complete landscape functioning model requires the use of many coupled models and therefore involves many formalisms (Quesnel et al., 2009)
- technological issues are also to take into account, such as interoperability between models and tools, software flexibility according to usage objectives, the need for high performance computational resources (for instance HPC resources), access to many data sources (Argent et al., 2006)
- due to the involvement of many disciplines, organisational issue are needed such as the definition of the interoperability framework, the share of the codes, the validation process of codes and coupled models, the share of documentation and information, the licenses attached to the software developments...

A few modelling frameworks or platforms have been developed in this way to deal with these issues over the last two decades (see Argent, 2004 for more complete review and historic) e.g. Spatial Modelling Environment (SME, Maxwell and Constanza, 1997), The Invisible Modelling Environment (TIME, http://www.toolkit.net.au/Tools/TIME) and hydrological derivative tools like E2 (Argent et *al*. 2009), OpenMI (http://www.openmi.org/, Moore and Tindall, 2005), and the Object Modeling System (OMS, Kralisch et al., 2005) that was used to apply the hydrological model J2000 (Fink et *al*. 2007). But these platforms have generally limited capabilities when integrating pluridisciplinary modelling approaches for landscape modelling (e.g. bio-physical processes, hydrology, agronomy, etc.) or when dealing with multi-scale issues in space (from field to regional scales) or / and in time (e.g. from seconds to years).

The OpenFLUID platform was recently developped for spatio-temporal model applications, providing a modern modelling framework, and allowing the implementation of single process components to be used and coupled. We describe hereafter the expected features for a collaborative modelling and simulation platform that guided the OpenFLUID development. The basic principles of the OpenFLUID platform and its main components are described, and we explain in more details how the problems of coupling models or model components are solved inside the system. As an example of the use of the software platform, the implementation of the hydrological model MHYDAS (Moussa et al., 2002) on one agricultural catchment is presented.

1. Expected features and concepts of a modelling and simulation platform

1.1. General requirements

Developing a multi-disciplinary modelling and simulation (M&S) framework or platform in environmental sciences, more particularly in integrative landscape functioning, requires both solid scientific background and the development of software tools. Based on the existing tools and also on our vision, we have briefly detailed the expected features and requirements hereafter.

From a scientific point of view, the inter-disciplinary approach required for landscape modelling increases the necessary amount of knowledge that we have to integrate in order to understand the dynamics of complex systems. Actual models or simulation tools are rarely explicitly based on a multi-formalism approach that facilitates the integration of heterogeneous models or coupling of models described in different formalisms. This is probably the first key factor for integrative landscape modelling.

A second key factor when dealing with landscapes, and therefore the heterogeneity and complexity of interconnected spatial structures, is a spatially distributed modelling approach that takes into account the specific elements of the landscape and the spatial patterns (such as anthropic terraces, ditches, embankments, …) that will influence or be modified by the bio-physical fluxes.

The combination of these first two key factors is a necessary condition for the development of models to be combined for an integrative landscape functioning modelling approach.

The software modelling framework should be developed to be flexible and easy to use by a variety of users (from student to expert scientist), allowing these users to develop new models or to use and reuse existing models within a common structure and a single software environment from development to simulation. It must be a software tool adapted to scientific research but also one that provides the software components for operational deployment and end-users applications. It should therefore propose well defined standards for development and data formats, and it should be in line with open source approaches.

Thirdly, a software platform is quite a large software development, and should therefore adopt a software engineering approach. It can take advantage of recent software technologies, improving maintenance and evolution processes with a quality objective, involving agile methods which are adapted to the scientific research context.

Finally, the modelling and simulation platform is also a gathering place for several communities, therefore sharing skills and knowledge about ontology, modelling approaches, model concepts and development, model use and study cases and information around the platform are key issues.

1.2. Specific concepts for modelling fluxes in landscapes

Besides the general requirements for a modelling and simulation platform, specific concepts are addressed and needed when dealing with fluxes in landscapes. This is particularly obvious with the coupling of bio-physico-chemical fluxes and with the simulation of these fluxes in different compartments of the landscape (both in a quantitative and qualitative aspects). There is a need to model the exchange fluxes between these compartments, with a sufficient degree of certainty (e.g. physically based modelling) and with as little constraint as possible to specific scales. One of the challenges is therefore to provide a conceptual representation of the landscape in line with the mathematical representation of the spatio-temporal processes, which will strongly influence the capabilities and performance of the model to represent the functioning of the landscape.

Among the numerical representations of the landscape, grid-based segmentation is the most commonly used. But the elementary pixels considered by grid-based landscape representations result in unsatisfactory solutions for representing irregularly-sized landscape features and gradients, especially in cultivated landscapes where many human-made features are to be considered as discontinuities for fluxes. Other methods like cell-based, contour-based or TIN-based segmentations of landscapes can be used. However, none of these landscape segmentations explicitly takes into account the man-made landscape features and they may therefore hinder simulations of the interactions and feed-backs between the landscape structures, the landscape processes and human activities.

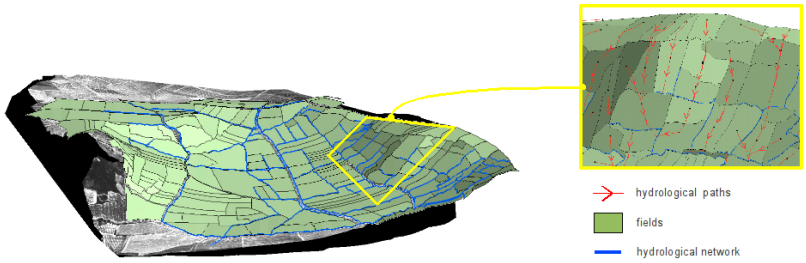

Figure 1: Numerical representation of the landscape with units and topology

As a possible solution, we developed a landscape discretization method that includes user-controlled delineated irregular, linear or areal units connected to each other along a tree-like structure (Figures 1 and 2). The result is a connected graph, where nodes are spatial units, and edges are connections between these units. Spatial units are categorized into several unit classes (for example for hydrological applications: surface units for plots, reach units for hydrological network - Lagacherie et al., 2010) according to the modelling objectives, the complexity and reality of the landscape and the available data and their precision (Fig 2 - step 1a). Once the classes and the units have been defined, the spatially oriented topology (connectivity between all units) is constructed (Fig 2 – step 2).

The model that would be applied on this numerical representation of the spatial domain is a composition of elementary simulation functions, where each one simulates variables on one or more classes of spatial units, in time at a given time step. The set of elementary functions composing the model is ordered in a specific way, ensuring that all variables needed for a specific function at a time step are already simulated before (i.e. at the same time step or at a previous one) by another function (Fig 2 - step 1b).

This first step of the modelling and simulation approach enables us to build a numerical representation in line with the coupled processes that we want to simulate. The next step consists in the parameterisation of the landscape representation, according to the applied fluxes model (Fig 2 – step 2). The final step is the simulation itself (Fig 2 – step 3).

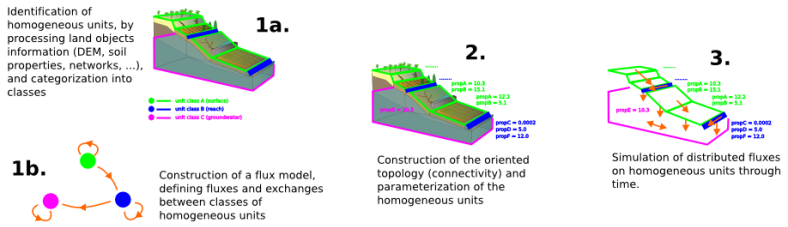

Figure 2: Step-by-step modelling and simulation approach and concepts for modelling fluxes in landscapes

2. Overview of OpenFLUID

2.1. The software platform

In order to set up a software support that implements the majority of the concepts and expectations previously mentioned, the OpenFLUID platform has been developed. It is composed of three major components (Figure 3): the simulation engine OpenFLUID-engine, the graphical user interface OpenFLUID-builder and the community web site OpenFLUID-web.

The OpenFLUID-engine component is the main component of the platform. It runs simulations using models, coupling fluxes and processes, applied to a numerical representation of the studied landscape. As this component brings the main concepts of the OpenFLUID platform, it is extensively described afterward (see 2.2).

The OpenFLUID-builder component is an adaptative graphical user interface for the OpenFLUID-engine. It gives the possibility to build and edit models, import landscapes representations, prepare and parameterise the simulation, run the parameterised simulation and exploit the results, with graphical capabilities.

Finally, the OpenFLUID-web component presents the OpenFLUID project and software. It is also a collaborative component, where users, modelers and developers can interact and exchange informations, ask questions, exchange pieces of source code, etc.

The platform has been mainly developped with C++ and Java programming languages, and relies on an object oriented design. It is available for Linux, Windows and MacOSX operating systems.

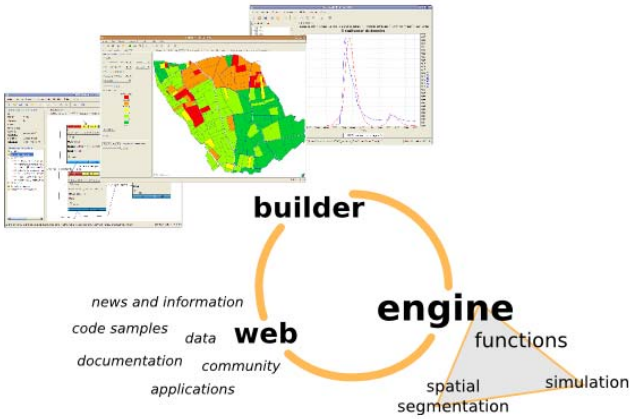

Figure 3: OpenFLUID platform and its components

2.2. Focus on OpenFLUID-engine

2.2.1. Main concepts and simulation features

As already mentioned, the OpenFLUID-engine component is the core component of the platform. It implements the concept of modelling fluxes in landscapes, as previously described (see 1.2): it runs simulation using coupled models applied to a numerical representation of the landscape.

The representation of the landscape is loaded into OpenFLUID-engine and integrated as a connected graph of spatial units, where nodes of the graph (the spatial units) are landscape elements, and edges of the graph are connections between these landscape elements. The connected graph will hold and share the values for the variables computed by the simulation function on each landscape element at every time step.

The coupled model is composed of an ordered list of simulation functions, exchanging values for variables in time and space. In order to check the data consistency of the coupled model, each simulation function declares the variables that will be produced, required or used (used only if present) for specific classes of spatial units, the required or used input data on spatial units and the needed function parameters. This makes up the signature of the simulation functions. Before each simulation, the signatures of the simulation functions

involved in a coupled model are combined in order to check the global consistency of the model at the data level.

During the simulation, the simulation functions are invoked at every time step, according to the coupled model definition. Although simulation functions are handling their own internal time steps, they are invoked at a fixed time step. They have to produce and share values for spatially distributed variables, according to their own signature. This is like an agreement between the engine and each simulation function.

The OpenFLUID-engine also provides non-functional features such as:

- the progressive output to disk during simulation, in order to free memory
- the ability to customize the output data and output formats
- the management of warnings and error messages
- the generation of simulation reports, with detailed information about the simulations
- some integrated utilities ("buddies") for automatic documentation of scientific background, dataset conversion, empty dataset generation.

2.2.2. Software architecture – technical points

Technically, OpenFLUID-engine mainly relies on a kernel-plugins software architecture, in order to build coupled models based on plugged computational codes that correspond to simulation functions. Thus, coupled models can be built using available simulation functions and adapted to the specific context and objectives of the simulation: simulated processes, spatial scales, available data. At runtime, simulation functions are dynamically plugged to the kernel according to the coupled model definition.

The simulation functions can encapsulate existing models, pieces of models, or can be developed "from scratch". Each modeller can develop their own simulation functions, as plugins to the OpenFLUID-engine kernel, in order to model and simulate the processes studied.

The kernel of OpenFLUID-engine does not contain any simulator, it is a kind of supervisor which manages data, manages plugged simulation codes, checks data consistency, monitors the simulation, and manages data inputs and outputs (Figure 4).

Inputs and outputs formats are standardized. The file formats rely on XML for a better integration of datasets and parameters, and also to guarantee inter-operability (Jolma and Rizzoli, 2003; Kokkonen et al., 2003) with third-party software and also OpenFLUID-builder.

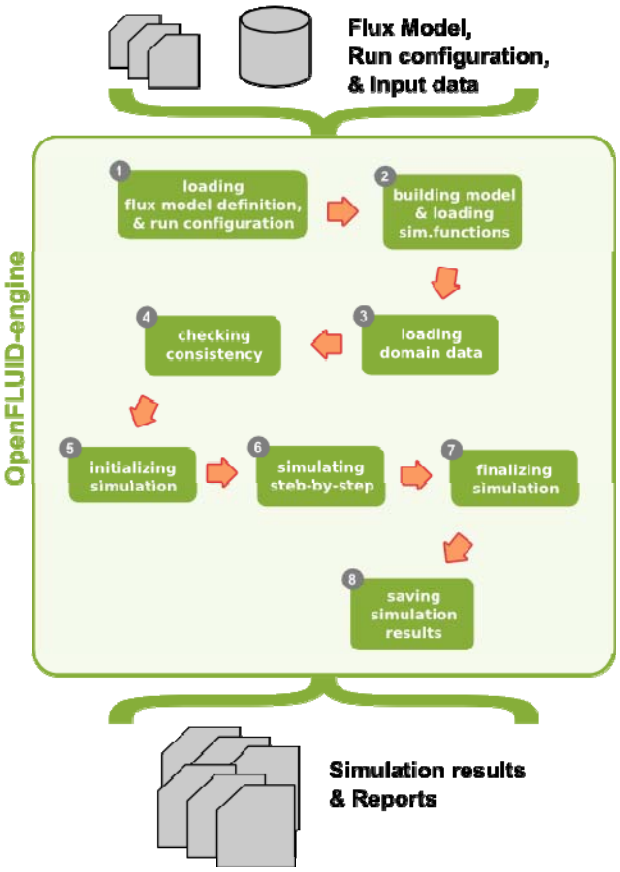

Figure 4: Internal functioning of OpenFLUID-engine

In order to develop and build the simulation functions, the OpenFLUID platform includes a SDK (Source Development Kit). This SDK is composed of the OpenFLUID-engine API libraries, documentation, and a specific extension for the Eclipse IDE. The OpenFLUID-engine has been developed in C++. Simulation functions are made of a C++ class, derivated from a generic superclass, so the computational codes encapsulated in simulation functions must be written in C++ compatible languages, such as Fortran, C or Python (Figure 5).

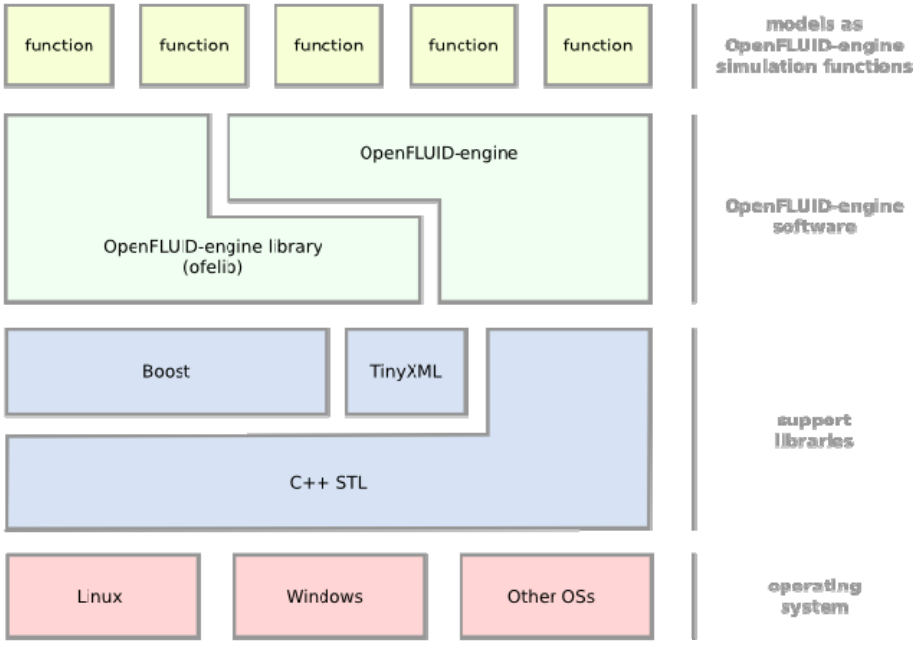

Figure 5: View of the OpenFLUID-engine software architecture,including the Application Programming Interface (API) of the OpenFLUID-engine library

2.3. OpenFLUID-builder

The OpenFLUID-builder component is the graphical user interface of the platform that interoperates with OpenFLUID-engine through XML data fluxes. It includes several functionalities and brings graphical support for the different modelling steps: from the choice of the model and the spatial domain to the visulation of the simulated results.

Using OpenFLUID-builder starts with the definition of a simulation project, combining a numerical representation of the landscape and a fluxes model that are compatibles. The numerical representation of the landscape can be defined in a preparatory phase through a dedicated data import. The model can be built with the model editor, by assembling different simulation functions in a consistent order.

Each simulation project in OpenFLUID-builder is based on a control panel, which guides the user through the different menus and provides help for parameterisation. To run a simulation via the graphical interface, the user has to define the period of simulation and time step for exchange of variables between simulation functions, model parameters, input data, distributed properties of each spatial unit, choice of variables to be backed-up in files at the end of the simulation, etc. The graphical interface helps to process the results and proposes time and space visualisations of all the produced variables.

The graphical interface can be easily extended through builder-extensions, which are plug-ins for OpenFLUID-builder. These extensions can concern either visualisation of data, data import and export and pre and post processing of the data.

3. Application example

As an application example, here is a simple scheme of a MHYDAS model (rainfall and infiltration-runoff partition, diffuse wave transfer on surface units and concentrated transfer in reach units), applied to a numerical representation of the landscape including 237 surface units and 372 reach units (Figure 6).

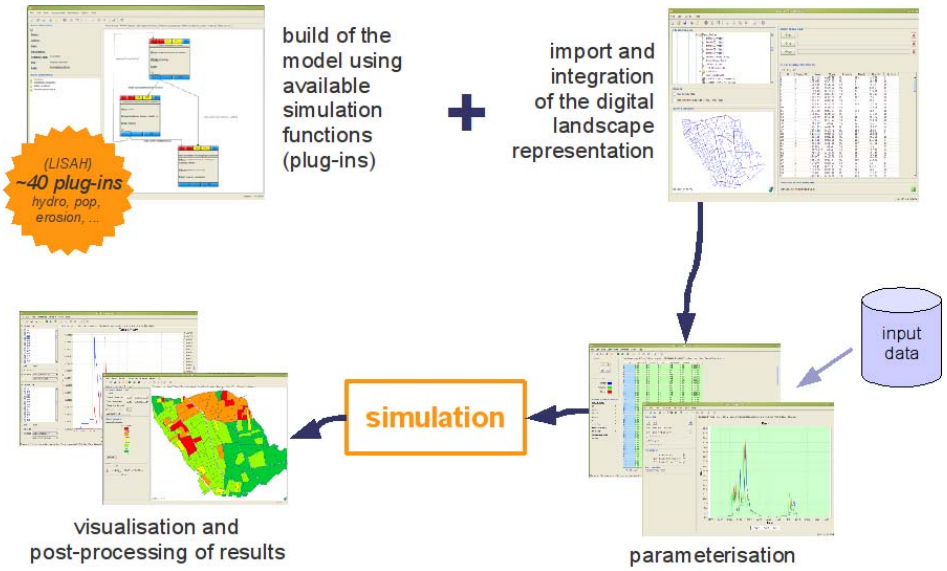

Figure 6: Example of a simulation project through OpenFLUID-builder, involving a MHYDAS model (surface hydrology module only)

The first step consists of building the coupled model using available simulation functions (currently forty functions have been developed at LISAH), and importing and integrating the representation of the studied landscape.

The next step is parameterisation of the simulation. This can be done using data extracted from various databases.

Once the simulation performed, results can be processed and visualised in space and time.

4. Software as a community builder

Beyond the software itself, the OpenFLUID platform was also designed to be a support for collaborative work on the modelling and simulation of fluxes in landscapes. Indeed, the OpenFLUID platform can help to build, organize and make a more dynamic community at the local scale or at a wider scale. This is enabled through the community approach that is linked to platforms in general, around tools and concepts such as sharing modelling concepts and approaches, the sharing of datasets and case studies and the sharing of source codes with re-use and for re-use. This mutually improves users and developers skills and knowledge. Information about the platform is continuously available, in order to give news and development status about the project, and also to supply reference documentation such as technical guides for installation, model development, user guides, scientific concepts on modelling and simulation, etc. The OpenFLUID-web component has been designed with this objective of community sharing and includes a major collaborative space (http://www.umr-lisah.fr/openfluid).

The collaborative work and community approach are also favoured by some of the technical features and choices for building the platform. For example, the plug-in approach for developing and using new simulation functions independently of the engine itself does not create a barrier for non experienced users. People can easily share their simulation functions, and use and test those shared by the community. Similarly, the adoption of a standardized input data formats and XML exchange format facilitates the sharing and use of study cases with the associated datasets.

As many actors are involved in the actual community (e.g. scientists, software engineers, students and few end-users), training sessions have also been given over the last 2 years.

Finally, the potential of the platform as a community software originates from the software applications and tools that were originally chosen for the development of all the components of OpenFLUID. It should be mentioned that the OpenFLUID platform has been developed using open-source approaches, promoting both interoperability, transparency and openness. Based on open-source software and solutions, specific software features have been developed and implemented for the development of simulation functions under the Eclipse IDE (OpenFLUID Eclipse plug-in), for the sharing of existing simulation functions through a common repository, for the automation of tests and for quality process on the OpenFLUID-engine and simulation functions, with user-defined tests and datasets, and for the automation of the documentation of each function.

Conclusion

The OpenFLUID platform is currently involved in scientific research either as a simulation tool for running well achieved models (like MHYDAS) with different datasets or for testing scenario, or as a framework for the development and test of new models. The OpenFLUID platform is also a support for higher education courses (about thirty students a year are trained). Due to its capabilities for time and space simulation, OpenFLUID has

been recently involved in two industrial partnerships in Research and Development programs dealing with water and environmental management.

OpenFLUID is mainly applied to environmental sciences (hydrology, soil and water contamination, erosion, sustainable agriculture, etc), but we think it can be used by other disciplines where processes are distributed in space and time.

In the future, the major OpenFLUID improvements will concern a) the ability to introduce a variable time step during simulation (in time and in space), b) the adjustment of our approach from coupling spatially distributed models to coupled spatial distribution of models. In both cases, the DEVS approach (Ziegler, 1976, 1987) seems to be a very promising approach.

References

Argent R.M., 2004. An overview of model integration for environmental applications--components, frameworks and semantics, *Environmental Modelling & Software*, 19, p. 219-234.

Argent R., Voinov A., Maxwell T., Cuddy S., Rahman J., Seaton S., Vertessy R. & Braddock R., 2006. Comparing modelling frameworks - A workshop approach, *Environmental Modelling & Software*, 21, p. 895-910.

Argent R., Perraud J.M., Rahman J., Grayson R. & Podger G., 2009. A new approach to water quality modelling and environmental decision support systems, *Environmental Modelling & Software,* 24, p. 809-818.

Fink M., Krause P., Kralisch S., Bende-Michl U. & Flügel W.A., 2007. Development and application of the modelling system J2000-S for the EU-water framework directive, *Advances in Geosciences, Copernicus Publications,* 11, p. 123-130.

Jolma A., & Rizzoli A., 2003. A review of interoperability techniques for models, data, and knowledge in environmental software. In Post, D. A. *(ed.), Modsim 2003: International Congress On Modelling And Simulation,* Vol 1-4, p. 1745-1750.

Kokkonen T., Jolma A. & Koivusalo H., 2003. Interfacing environmental simulation models and databases using XML, *Environmental Modelling & Software,* 18, p. 463-471.

Kralisch S., Krause P. & David O., 2005. Using the object modeling system for hydrological model development and application. *Advances in Geosciences*, 4, p. 75-81.

Lagacherie P., Rabotin M., Colin F., Moussa R. & Voltz M., 2010. Geo-MHYDAS: A landscape discretization tool for distributed hydrological modeling of cultivated areas. *Computers & Geosciences*, in press.

Maxwell T. & Costanza R., 1997. A language for modular spatio-temporal simulation, *Ecological Modelling,* 103, p. 105-113.

Moore R.V. & Tindall C.I., 2005. An overview of the open modelling interface and environment (the OpenMI), *Environmental Science & Policy, Research & Technology Integration in Support of the European Union Water Framework Directive,* 8, p. 279-286.

Moussa R., Voltz M. & Andrieux P., 2002. Effects of the spatial organization of agricultural management on the hydrological behaviour of a farmed catchment during flood events, *Hydrological Processes*, 16, p. 393-412.

Quesnel G., Duboz R. & Ramat É., 2009. The Virtual Laboratory Environment - An operational framework for multi-modelling, simulation and analysis of complex dynamical systems. *Simulation Modelling Practice and Theory*, 17, p. 641-653.

Zeigler B., 1976. *Theory of Modeling and Simulation* (first ed.). Wiley Interscience, New York,.

Zeigler B., 1987. Hierarchical, modular discrete-event modelling in an object-oriented environment, *Simulation,* 49, p. 219-230.