Model Driven Engineering, applied to observation problems of socio-environmental models

Hasina Lalaina Rakotonirainy
EDMI, University of Fianarantsoa
Fianarantsoa, Madagascar
hasina.rakotonirainy@cirad.fr

Jean-Pierre Müller
GREEN, CIRAD
Montpellier, France
jean-pierre.muller@cirad.fr

Bertin Olivier Ramamonjisoa
EDMI, University of Fianarantsoa
Fianarantsoa, Madagascar
bertinram@yahoo.fr

Abstract

Researchers need to use complex models to understand socio-ecosystems (SES). However, observing SES models becomes a difficult task and no general framework exists to solve this problem. The aim of this paper is to propose a generic framework to specify the observation of SES models in order to easily generate the indicators that thematicians wish to monitor during simulation. To that end, we propose to reformulate the observation process as a transformation problem between data structures. This enables the concepts of Model Driven Engineering (MDE) to be used in the implementation of the domain specific languages (DSL). We designed dedicated languages to enable specify of what to observe, define which observation strategy to use, and decide how to generate the indicators.

I. Introduction

Research on the sustainable management of renewable natural resources (RNR) has shown how difficult it is to understand the interactions existing between social dynamics and biophysical dynamics characterizing a given socio-ecosystem (SES). Some increasingly complex models have been developed to facilitate the shared understanding of stakeholders directly or indirectly involved in local RNR management. However, using complex models creates the need to observe what happens during the simulation [1] in terms of indicators relating to issues of model comprehension, research or development. Observing simulation thus becomes as important as the simulation itself. Observation consists of ascertaining the state and the evolution of the model during simulation [2] and establishing synthetic indictors for them.

To that end, some modelling and simulation tools such as MIMOSA [3] and JAMES II [4] implement the observation design model which signals changes in model state. Some tools such as OSIF [5] use aspect-oriented programming. Other tools such as PowerDEVS [6] use the classic reactive or proactive mechanism of DEVS formalism. When simulating discrete events, some researchers have proposed to extend PDEVS formalism to a generic observation mechanism [2] not influencing model behaviour during simulation. Despite these efforts, the approaches proposed for model observation are relatively ad hoc. They depend on the formalism used during the model construction and a general framework has yet to be designed.

The purpose of this paper is to propose a generic framework to specify the observation of SES models in order to easily generate indicators. To that end, we propose to reformulate the issue of model observation as a problem of transformation of data structures into other data structures. Thus, specifying model observation aims at describing which structures of the model need to be observed, with what observation strategy, then specifying a transformation chain to transform these structures into other data structures that are the indicators sought by thematicians.

With this formulation, model driven engineering (MDE) concepts [7] can be used to implement domain-specific languages (DSL) that are dedicated to observing complex models.

In section II, we introduce and justify the methodology used to design the observation DSLs. After that, we present the observation process we propose for SES models (section III), along with the preliminary results (section IV). Lastly, we provide a glimpse of some future work in the conclusion.

II. Methodology

MDE is a methodology to design and develop for model-based software packages [7] that enables easy construction of DSLs. One of the advantages of using DSLs is that modellers can focus on research concerns and fields. The application code is then generated automatically. For DSL construction, OMG (Object Management Group)[8] proposed the modelling pyramid (Figure 1) which is based on five basic concepts: the real system, the model allowing to represent that system, the meta-model which is use to define the model, the meta-meta-model that is dedicated to specify the meta model and the transformations between models. In addition, we showed in [9] that by using the concepts proposed by MDE, we are able to formulate the problem of initializing and observing complex socio environmental models in a generic manner using DSLs.

In order to be able to use MDE, we need to formulate the observation process in terms of MDE (Figure 2), specifying the meta models to describe the part of the models simulation trajectories we wish to observe (the observables - L4), the structure of the indicator we wish to construct (L6) and transformation between the observables and the indicators (L5), bearing in mind that L1 to L3 [9] are dedicated to initialization.

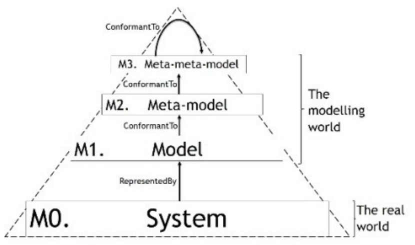


Fig. 1. OMG modelling pyramid

III. Observation process

Figure 3 shows the overall stages we propose to construct the indicators desired by thematicians based on the data structures of a SES model. In fact, during simulation, it is possible to track changes in the data structures of the model by defining an observation strategy and obtaining trajectories. Constructing a trajectory may require us to specify a sampling strategy. Trajectories obtained in that way are time-indexed data structures (sequence <dates, data structures >). Consequently, we need to further transform these time-indexed data structures (i.e. the trajectories) into manipulable data structures in order to obtain indicators. Lastly, the indicators can be stored in a data medium, or visualized.

IV. Preliminary results

We specified the meta-models of the domain specific languages (DSL) L4, L5 and L6 with the Ecore meta-meta-model of the EMF (Eclipse Modelling Framework) [10]. These DSLs are used to specify:

- The part of the model we wish to observe during the simulation (L4) and how to form a set of trajectories
- The indicators with their visual presentations (L6)
- The transformations of the trajectories into indicators (L5).

A. Language L4

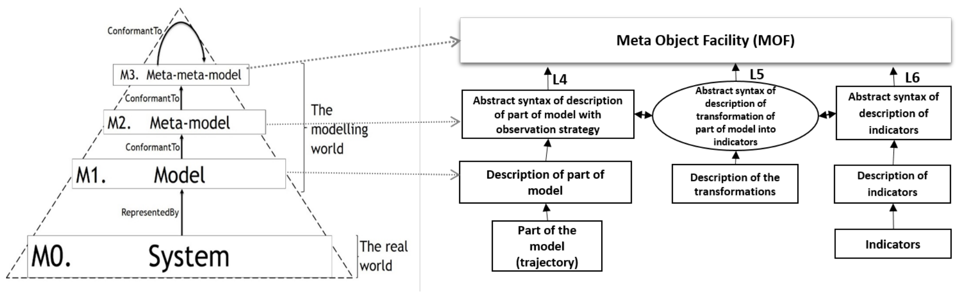
Language L4 is designed to specify the part of the model that we wish to observe and track during simulation with a certain observation strategy.

1) State of the art for model observation: For each simulation of a SES model, such as the Mirana model [1], the data and interactions between the elements of the model are numerous and very complex. Tracking everything that takes place in the model is very demanding in terms of time and in storage space. To solve the running time problem, some tools such as SimExplorer [11] or OpenMOLE[12] are able to distribute the simulations of a model with sets of different parameters on clusters of machines. Some computer programs use the lazy evaluation technique [13] and avoid computing pointless results when running a program. However, the solution we propose needs to be adapted within thematicians needs. In order to specify a part of a model, we need to provide a way of precisely selecting and sampling any part of the state of the model at different levels and on different scales (spatial and temporal). The most common way in IT is to use a query language such as SQL to query relational databases, OQL for object-oriented databases, OLAP for multidimensional databases, XPath to browse DOM or XQuery for XML data. Nevertheless, a query language depends on the data structures to be queried. In the literature, XML is acknowledged as the universal data description format [14] and its tree structure can be used to represent the hierarchy of any data structure. Thus, in order for L4 to be as generic as possible, we propose to create an abstract syntax based on XPath and XQuery in order to specify what we want to observe in the trajectory of the model. However, we do not work on XML or database content, but on the data structures themselves. The fact that a trajectory has to be generated rather than a state means that we have to add a sampling strategy.

2) Observation strategy: Defining an observation strategy consists of specifying how, when and how often the model is observed during simulation, in order to obtain trajectories. In fact, two types of observation strategy can be specified, namely:

- $O_{observer}$: Each element of the model will signal its change in state to a registered observer using the observer design model [15]. In this case, the simulation model has to implement this type of pattern and it is necessary to describe the functions that enable an observer to subscribe to the element one would like to track and how the change in state is signalled.
- O_{timed}: An observer queries the state of an observable element of the model, specifying the observation dates (unique, at constant time steps, etc.). It is then necessary to specify how to extract the information (*getters*).

The samples of the model will then be extracted during simulation according to the choice of observation strategy to form trajectories. For example, in the context of SES models, it is possible, by specifying an observation strategy, to track changes in the quantity of resource stocks for a given geographical space.



ConformantTo
M3. Meta-meta-model
ConformantTo
M2. Meta-model
ConformantTo
M1. Model
RepresentedBy
M0. System
The modelling world
The real world
Meta Object Facility (MOF)
L4
L5
L6
Abstract syntax of description of part of model with observation strategy
Abstract syntax of description of transformation of part of model into indicators
Abstract syntax of description of indicators
Description of part of model
Description of the transformations
Description of indicators
Part of the model (trajectory)
Indicators

Fig. 2. Correspondence between the modelling pyramid and the observation process

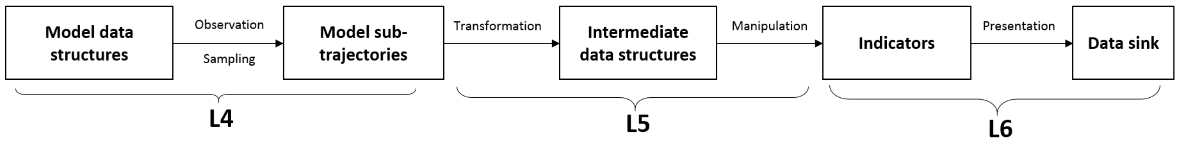


Fig. 3. Indicator construction process

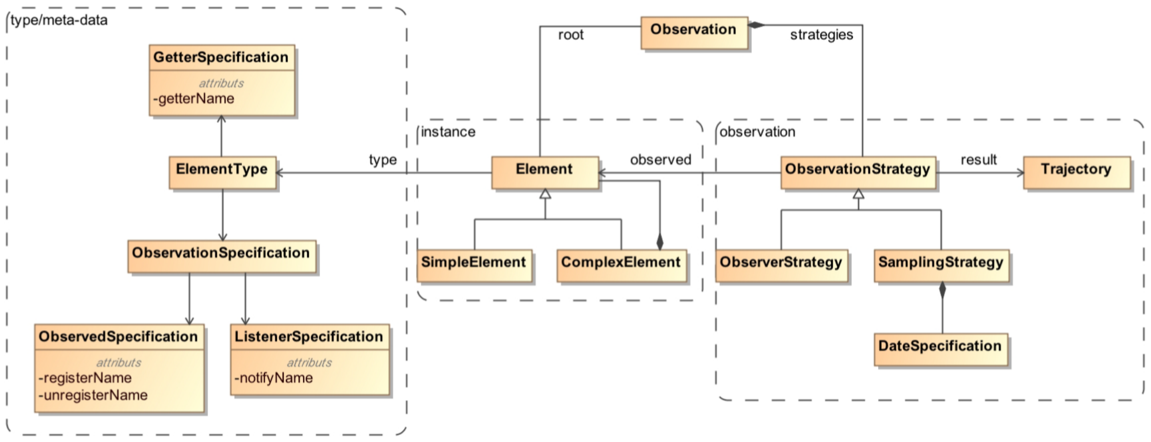


Fig. 4. Extract of the language L4 meta-model

An extract of the language L4 meta-model is given in Figure 4.

B. Language L5

Language L5 is used to transform the trajectories derived from the implementation of language L4 into indicators sought by thematicians (specified from language L6 section IV-C). However, the trajectories arising from simulation are obtained incrementally in line with model evolution and the observation strategy defined from language L4. It then becomes necessary to have a meta-model capable of managing the data flows in order to obtain indicators from the data sequences arising from the simulation. For that purpose, we propose to employ "DataFlow" architecture [16], which is an architecture and technique that has been widely used in the field of parallelism since the 1970s to perform transformations that are both incremental and parallel, along with the data stream management system (DSMS) [17] which can be used to manage and query data in a continuous data stream.

Thus, to obtain indicators, we propose the process described in Figure 5. The principle consists of recovering the trajectories generated incrementally during simulation, then preprocessing them with a view to extracting the temporal part, in order to adapt the format of the data to data stream handling. The next stage consists of implementing a transformation chain, specified by the user, on the data stream for as long as the data structures to be manipulated are available. The results are modified on each arrival of new data structures.

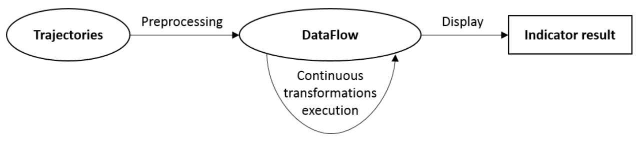


Fig. 5. from trajectories to indicators

Two types of processing are therefore taken into account, namely:

- Preprocessing
- Manipulation of data structures to construct indicators.

1) Preprocessing: Preprocessing makes it possible to obtain some computer data structures from time indexed data sequences, which are recovered during simulation. In substance, for each available time indexed data item (simple or complex), an attributive structure is created with a time attribute, along with an attribute with the simple or complex data item.

2) Manipulation of data structures to generate indicators: In order to understand the relations existing between the data structures of a model, or obtain synthetic indicators, thematicians use data processing and statistical analysis. They call upon tools such as R, STATA, SPSS or SAS. These tools have several packages able to automatically process various calculations, methods and statistical models. When simulating a SES model, the data structures available derived from the transformation of trajectories, along with the indicators to be constructed, evolve in line with the evolution of the model. Consequently, the indicators obtained from various operations (mean, median, standard deviation, quantile, etc.) also evolve in line with the data structures available during the simulation process.

We therefore propose a meta-model (Figure 6), based on data stream management, which meets the expectations of thematicians, making it possible to specify transformations (statistical methods, arithmetic calculations, etc.) along with the types of data they need in order to function, so as to execute them and obtain some results each time the necessary data are available.

Specification based on this meta-model is used to obtain a graph where the nodes are the transformations to be carried out and the arcs serve for the circulation of the available data streams, in order to form the inputs needed to activate a node, and their outputs (results obtained after the performance of a transformation), which may be indicators sought by thematicians. The outputs of a node can be used as inputs for other nodes. The resulting graph represents the execution structure for the transformation chain that needs to be implemented to generate the desired indicators.

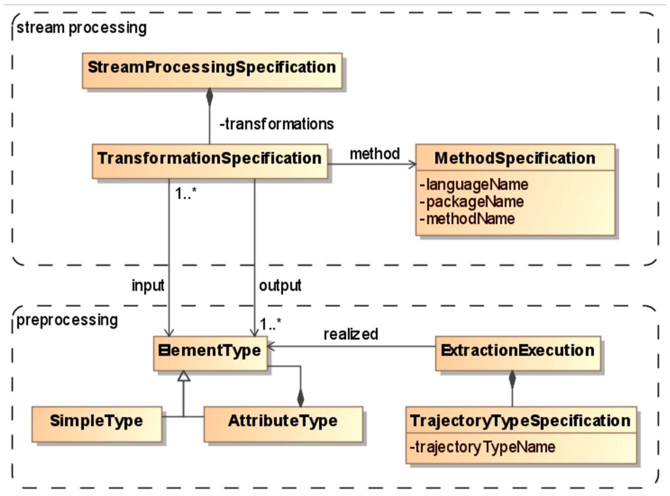


Fig. 6. L5 meta-model to obtain the indicators

For example, let us assume that, during simulation, we wish to ascertain changes in the average area of a type of specific habitat, in hectares, within a geographical space. To do that, we merely need to specify a function that is able to calculate the mean of a sequence of data, along with the input data stream needed for its execution. Each time the simulation model evolves (i.e. new data available), the function uses the new available input data structure and updates the mean of the values observed up to then.

C. Language L6

Language 6 serves to specify the structure of the indicators, along with their presentations on the user side. It should be noted that an indicator is a computer data structure that represents the information needed by thematicians to understand the functioning of the model so as to carry out an assessment or make decisions regarding the system being studied. The indicator may be stored in a data medium (database, files, etc.) for possible uses, or visualized in an interactive visualization tool.

For this, we propose a meta-model comprising two parts:

- A part to specify the structure of the indicators
- Another part to specify data sinks charged to visually present the indicators or store them on a medium.

A part of the meta-model of language L1 [9] is used to specify data structures. It can also be used to describe the structures of indicators in language L6.

1) State of the art for the visual presentation of data: Many researchers have been interested in the visualization problem for around forty years. Solutions for representing data, information or knowledge are continuing to evolve. In the 1970s, some computer science laboratories worked on the visualization of scientific data, which led to the development of metric and statistical methods, techniques and processes to represent scientific data. In the 1990s, multiple-expert cross-referenced approaches bringing together engineering and cognition were proposed in order to improve the understanding of the data being studied. These approaches gave rise to an important field of research on the graphic representation of information known as ”InfoVis” [18]. It consists of transforming data structures into visual forms that can be understood by stakeholders in the system. To that end, Card et al. [19] formalized the data visualization process (Figure 7) in order to enhance knowledge starting from raw data or data tables.

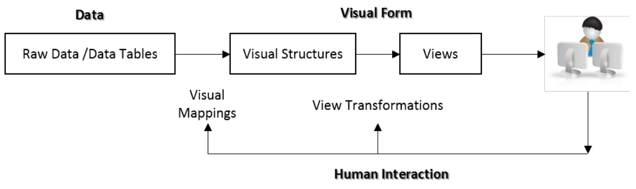


Fig. 7. Process of creating visualization according to [19]

At the same time as these approaches, some other data visualization techniques saw the light of day. Among others, the technique called ”design” or ”OOR” (for Object-Oriented Representation), inspired from the object-oriented concept, can be used for the visual representation of complex data. The so-called ”extended viewpoint” approach [20] was developed to represent and control the visualization of complex information systems (CIS). To that end, Bihanic and Polacsek [20] used MDE and defined DSLs composed of a modelling language conforming to MOF [21] to represent the data of the CIS (source language), a language called VSML (Visual Semantic Unified Modelling Language) also conforming to MOF to represent graphically the model of the CIS (target language) and a model transformation language used to pass from a CIS model to a visual representation model (Figure 9).

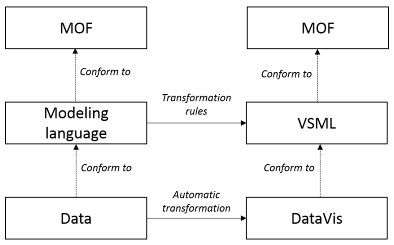


Fig. 9. Transformation of the modelling language to the VSML

In computer science, as in scientific visualization, there arose the need to separate what one wishes to visualize (the model) from how one visualizes it (the view). To that end, Trygve Reenskaug developed the Model View Controller (MVC) architecture [22], which is a standardized framework whose principle is to separate the data (model), the presentation (the view) and the processing (the controller) for easier handling, and easier visualization of voluminous and complex data.

For the visual presentation or storage of indicators, we based ourselves on [20] and [22] and we propose a meta-model (Figure 8) based on the ”Model View” (MV) architecture of the MVC design model. The MV architecture consists of distinguishing the data model, notably the indicators and their visual presentation on the user side, or their storage on a data medium.

2) Representation of indicators: The proposed meta-model enables the user to specify:

- *A data model*: to specify the class of model, along with the specific methods adapted to loading and containing the indicator data
- *A view (or data medium)*: to describe what component (e.g. list, tree, database table, file, etc.) will take charge of visualizing or storing the data recovered from the data model
- *The association between the model and the view*: to specify the method (setModel) used to associate the data model with one or more views in order to generate one or more visualizations of the indicators. A data model can be visualized or stored at the same time in different components. This specification then enables the automatic generation of an executable code to display the data via a tree, plot, map, label, etc. or to store them on a data medium such as the tables of a database, files, etc..

3) Example of using L6: Let us take the example of a data structure (or an indicator) that describes the area of habitats in hectares in a geographical space at a given time. To display these data in a graphic component ”JTable” of the Java language, it is necessary to specify:

- The model class (DefaultTableModel) along with the parameters of its constructor or the methods associated with it (addColumn, addRow, etc.) in order to load the data into a table model instance
- The class of the graphic component (JTable), to display the data in a table
- The relation (setModel) between the view (JTable) and the model (DefaultTableModel) so that the data recovered from the DefaultTableModel model are displayed in the JTable component. This specification automatically generates the executable code, making it possible to visualize the data in a table (Figure 10).

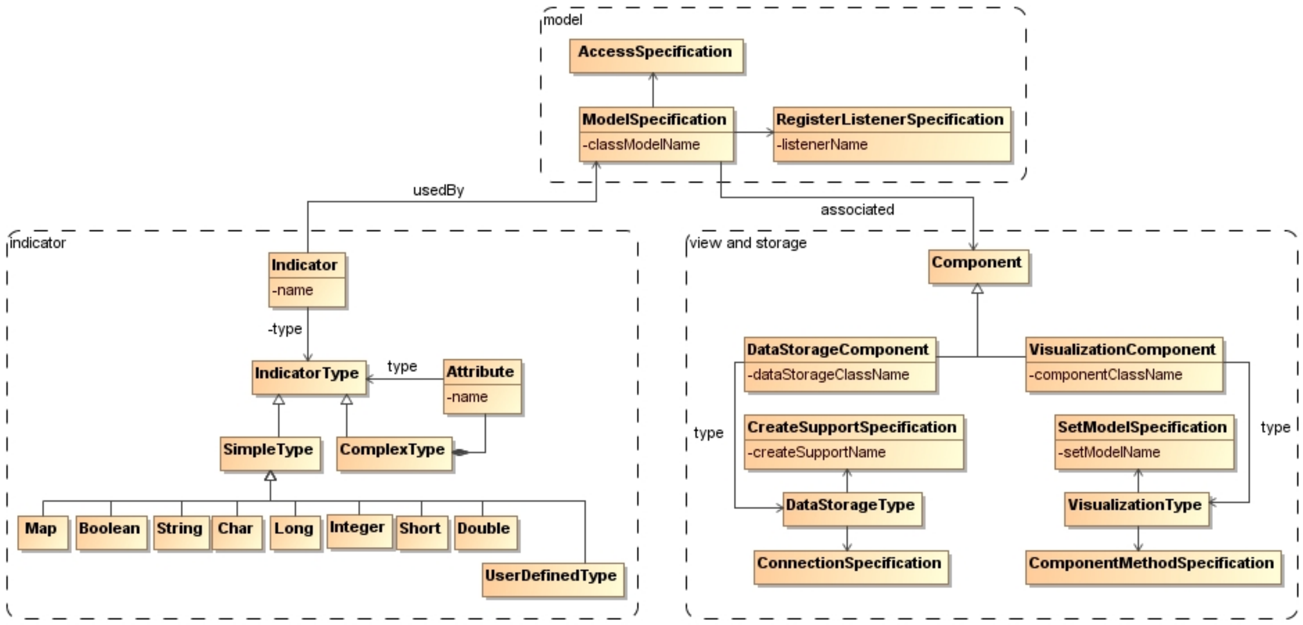
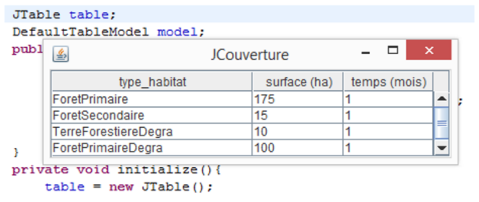


Fig. 8. Meta-model of language L6



```
JTable table;
DefaultTableModel model;
publ
}
private void initialize(){
    table = new JTable();
```

JCouverture

type_habitat	surface (ha)	temps (mois)
ForetPrimaire	175	1
ForetSecondaire	15	1
TerreForestiereDegra	10	1
ForetPrimaireDegra	100	1

Fig. 10. Example of visualization generated from L6

V. Conclusion and Prospects

In this paper, we have specified DSLs L4, L5 and L6 using MDE concepts making it possible to formulate the observation process for complex SES models in a generic manner. Nevertheless, in order for this proposal to be operational, we still need to specify, for each meta-model, a textual or graphical concrete syntax, along with generation of the associated code.

Acknowledgements

This research was funded by a PhD grant from AIRD/CIRAD.

References

[1] S. Aubert, J.-P. Müller, and J. Ralihalizara, “Mirana: a socio-ecological model for assessing sustainability of community-based regulations,” *International Environmental Modelling and Software Society (iEMSs)*, p. 8, 2010.

[2] G. Quesnel, R. Trepos, and r. Ramat, “Observations of discrete event models.” in *SIMULTECH*, N. Pina, J. Kacprzyk, and M. S. Obaidat, Eds. SciTePress, 2012, pp. 32–41.

[3] J. P. Müller, “Mimosa: using ontologies for modeling and simulation,” in *Proceedings of the 8th Asia-Pacific complex systems conference (Complex’07)*, 2007.

[4] J. Himmelspach, “James ii: Extending, using, and experiments,” in *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS ’12. ICST, 2012, pp. 208–210.

[5] J. Ribault, O. Dalle, D. Conan, and S. Leriche, “Osif: a framework to instrument, validate, and analyze simulations,” in *Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2010, p. 56.

[6] F. Bergero and E. Kofman, “Powerdevs: a tool for hybrid system modeling and real-time simulation,” *SIMULATION*, vol. 87, no. 1-2, pp. 113–132, 2011.

[7] J.-M. Jézéquel, B. Combemale, and D. Vojtisek, *Ingénierie Dirigée par les Modèles : des concepts à la pratique...*, ser. Références sciences, Ellipses, Ed. Ellipses, Feb. 2012.

[8] OMG, “About the Object Management Group,” 2014, 05.10.2014. [Online]. Available:

http://www.omg.org

[9] H. L. Rakotonirainy, J. Müller, and B. O. Ramamonjisoa, "Towards a generic framework for the initialization and the observation of socio-environmental models," in *Model and Data Engineering - 4th International Conference, MEDI 2014, Larnaca, Cyprus, September 24-26, 2014*, 2014, pp. 45–52.

[10] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework 2.0*, 2nd ed. Addison-Wesley Professional, 2009.

[11] F. Chuffart, N. Dumoulin, T. Faure, and G. Deffuant, "Simexplorer: Programming experimental designs on models and managing quality of modelling process," *IJAEIS*, vol. 1, no. 1, pp. 55–68, 2010. [Online]. Available: http://dx.doi.org/10.4018/jaeis.2010101304

[12] J. Passerat-Palmbach, M. Leclaire, R. Reuillon, Z. Wang, and D. Rueckert, "OpenMOLE: a Workflow Engine for Distributed Medical Image Analysis," in *International Workshop on High Performance Computing for Biomedical Image Analysis (part of MICCAI 2014)*, Boston, United States, Sep. 2014.

[13] T. Johnsson, "Efficient compilation of lazy evaluation," *SIGPLAN Not.*, vol. 39, no. 4, pp. 125–138, Apr. 2004.

[14] D. Florescu and D. Kossmann, "Storing and querying xml data using an rdmbs," *IEEE Data Engineering Bulletin, Special Issue on*, vol. 1060, no. 22, p. 3, 1999.

[15] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-oriented Software*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1995.

[16] J. B. Dennis and D. P. Misunas, "A preliminary architecture for a basic data-flow processor," *SIGARCH Comput. Archit. News*, vol. 3, no. 4, pp. 126–132, Dec. 1974. [Online]. Available: http://doi.acm.org/10.1145/641675.642111

[17] R. Motwani, J. Widom, A. Arasu, B. Babcock, S. Babu, M. Datar, G. Manku, C. Olston, J. Rosenstein, and R. Varma, "Query processing, resource management, and approximation in a data stream management system." CIDR, 2003.

[18] K. Andrews, S. F. Roth, and P. C. Wong, Eds., *IEEE Symposium on Information Visualization 2001 (INFOVIS'01), San Diego, CA, USA, October 22-23, 2001*. IEEE Computer Society, 2001.

[19] S. K. Card, J. D. Mackinlay, and B. Shneiderman, Eds., *Readings in Information Visualization: Using Vision to Think*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999.

[20] D. Bihanic and T. Polacsek, "Models for visualisation of complex information systems," in *16th International Conference on Information Visualisation, IV 2012, Montpellier, France, July 11-13, 2012*, 2012, pp. 130–135. [Online]. Available: http://dx.doi.org/10.1109/IV.2012.32

[21] MOF, "Meta object facility (mof) 2.0 core specification," OMG, Tech. Rep. formal/06-01-01, 2001, oMG Available Specification. [Online]. Available: http://www.omg.org/cgi-bin/doc?formal/2006-01-01

[22] S. Burbeck, "Applications programming in smalltalk-80(tm): How to use model-view-controller (mvc)," 1987.