TerraME GIMS: An Eclipse Plug-In for Environmental Modeling

Tiago Lima, Tiago Carneiro
TerraLAB (www.terralab.ufop.br)
Federal University of Ouro Preto
Ouro Preto, Brazil
tiago@decea.ufop.br, tiago@iceb.ufop.br

Sergio Faria
Federal University of Minas Gerais
Belo Horizonte, Brazil
fariamaracai@yahoo.com.br

Pablo Silva, Miguel Pessoa
LEDS (www.leds.ufop.br)
Federal University of Ouro Preto
Ouro Preto, Brazil
{pablosufop, guelmisp}@gmail.com

***Abstract*—TerraME is a platform for modeling and simulation of environmental systems that offers a conceptual basis and services to build environmental models through a high-level programming language called Terra Modeling Language. However, the use of a programming language is still a limiting factor since its main users are researchers with different backgrounds who usually lack basic knowledge of algorithms and programming techniques. So, this work presents the development of TerraME GIMS, an Eclipse plug-in for environmental systems modeling through visual metaphors that graphically represent the model.**

***Index Terms*—Visual programming, graphical user interface, environmental modeling, TerraME GIMS, Eclipse, plug-in**

I. Introduction

The Earth system comprises interaction between socio-economic systems (of anthropic origins as land use system) and biophysical systems (of natural origins as the atmospheric system). In general, Earth system phenomena have a complex nature like the land use and land cover change process. It requires the use of modeling and simulation tools to study, understand and represent the phenomenon. Besides, for dealing with these problems it is necessary a multidisciplinary team of specialists from different fields of knowledge. One of the major challenges is to make explicit the different conceptions that each person has about a phenomenon behavior and about the model conception and design. Sophisticated user-friendly computational platforms are required for dealing with this problem.

Modeling involves building a simplified representation of the reality. It helps to clearly define problems and concepts, and provides means to analyze an observed behavior representing it in a synthetic environment and reporting simulation outcomes [1]. Modeling and computer simulation have been used in scientific researches to address problems of complex nature, when the solution has a high cost or cannot be obtained through experiments [2]. Therefore, these methods and tools are essential to study terrestrial systems behavior, which are typically represented as spatial dynamic models that describe spatial patterns of changes evolving over time [3].

The choice of using a particular tool for environmental modeling can be based on features it offers (e.g. integration with databases, scientific visualization, modeling language), on the theory or paradigm on which it is based - System Dynamics [4] (Vensim, Stella, Simile), Agent Theory [5] (Swarm, Repast, NetLogo), Cellular Automata [6] (TerraME), Discrete Event System Specification [7] (JDEVS, CD++Builder). In this work, TerraME platform was chosen for the following factors: (i) it allows building models in multiple scales, (ii) it supports building multi-paradigm models, (iii) it provides scientific visualization services, (iv) it offers integration with geographic databases, and (v) it provides a high-level modeling language.

TerraME (Terra Modeling Environment) [8] is a software platform for modeling and simulation of environmental phenomena. It allows building dynamic spatial models integrated to a Geographic Information System (GIS). Through Terra Modeling Language (TerraML), it offers a high level programming language which allows the modeler (user) to represent data structures and rules that govern the models behavior more clearly and efficiently than using general purpose programming languages, like C++ or Java.

However, despite TerraML facilitates the representation of spatial dynamic models, the assimilation of its language concepts and buildings still presents high level learning difficulty. It still demands some programming skills from the user due to the inherent complexity of the concepts that it implements and of environmental phenomena to which it applies. Thus, professionals and researchers that are not familiarized with algorithms and programming techniques present difficulties in its use. This is the case for most of the specialists who are studying environmental systems like geographers, ecologists, biologists, sociologists, economists. However, they are the most interested and involved in modeling of environmental systems and they have knowledge about application domain.

Therefore, a new and higher abstraction level is needed allowing to focus on solving problems in the application domain. The representation of models through graphical components as diagrams, instead of algorithms, should make the process of building environmental systems models more intuitive and efficient. It should also increase the productivity of current users and decrease difficulty and learning time for new users.

This work presents the development of TerraME GIMS (TerraME Graphical Interface for Modeling and Simulation), a plug-in for the Eclipse platform which allows users to build environmental models through a graphical description of these, and the TerraML source code is automatically generated.

978-1-4673-6288-7/13 © 2013 IEEE

TOPI 2013, San Francisco, CA, USA

II. TerraME Platform

TerraME (Terra Modelling Environment) [8] is a software platform for modeling and simulation of environmental phenomena. Under continuous development, it is currently maintained by TerraLAB [9] and freely available for download (http://www.terrame.org/). It provides services for describing and simulating spatial dynamic models integrated to a Geographic Information System (GIS).

Several studies have used TerraME as a platform for modeling and simulation. [10] developed a model for spatial games and presented results demonstrating how mobility affects the Nash's equilibrium. [11] made an analysis over the spatial relationships between objects at different scales and implemented through TerraLib two types of relationships - hierarchical and network-based. The platform was also used by [12] in a case study where it was analyzed how the existence of different rules of land use affects the landscape dynamics at regional level.

TerraME was built based on layered architecture, where the lower layers provide functionalities over which higher layers are implemented, as illustrated in Figure 1. On the lower layer, TerraLib provides services for management and analysis of spatiotemporal data. On the second layer, TerraME Modeling Framework provides services for simulation, calibration and validation of models that can be used through C++ programming language. However, its application programming interface (API) has a complex syntax and is therefore difficult to use for people without a great knowledge of programming techniques. The next layer is formed by the interpreter and runtime environment of TerraME Modeling Language, which extends the LUA programming language by adding new data types designed for dynamic spatial modeling and services for simulation and evaluation of models. Finally, application layer is composed of the models developed by users.

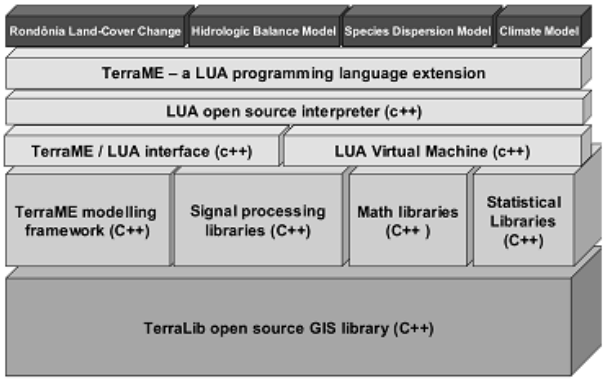

Fig. 1. TerraME architecture [8]

The conceptual design implemented by TerraME follows the scale concept [13]. Thus, an environment on earth can be described by a synthetic (virtual) environment where analytical entities (rules) change spatial properties over time. Therefore, describing a phenomenon as a model is done by representing its behavior in time and space.

TerraME offers data structures and services to represent spatial, temporal and behavioral models, as well as to construct models with multiple scales. Terra Modeling Language (TerraML) permits describing spatially dynamic models through the following data types: (a) Environment: to represent the scale concept and to enable the development of models with multiple scales; (b) CellularSpace, Cell, Neighborhood: to represent the space, their properties and topological relations; (c) Agent, Automaton, State, Jump, Flow and Trajectory: to represent the behavior of system; (d) Timer, Event and Message: to represent time, defining the instant and execution order of the events. Also, it is possible to visualize the model at runtime and save results using the Observer data type.

So, in this work we developed a new layer to TerraME software architecture, which provides a higher abstraction level for users. The Eclipse platform is used as base to develop the graphical user interface (GUI).

III. Eclipse-based Development

Developing a computer system and a computational model for socio-environmental phenomena are essentially similar activities. Therefore, the use of an integrated development environment (IDE) with features that support software development process is also desirable and applicable on environmental models building. However, creating a software application of this nature has a very high cost. A better alternative than building a new IDE from scratch is to use existing platforms, tools and frameworks.

Eclipse [14] is an open source platform that provides the basis to develop tools and applications based on IDE. Its integration capability achieved by plug-in based architecture (Figure 2) is a great advantage. A plug-in is the smallest functional unit that can be developed and distributed separately. So, new applications are developed extending the Eclipse platform via plug-ins [14].

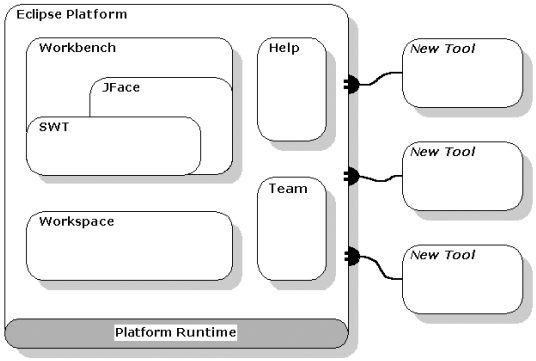

Fig. 2. General overview of Eclipse SDK [14]

Another advantage of using the platform are the several frameworks available that facilitate the development of software and plug-ins. Particularlly, on development of TerraME GIMS, the frameworks Eclipse Modeling Framework, Graphical Editing Framework and Graphical Modeling Framework

were used. The tools EuGENia [15] and SWTBot [16] were also used. It was also used a third party plug-in for highlighting syntax of TerraML source code.

The Eclipse Modeling Framework (EMF) is a modeling framework for building tools based on a data model. From a model specification written in XMI (XML Metadata Interchange), it provides tools and runtime support to produce a set of Java classes for the model, a set of adapter classes that enable viewing and command-based editing of the model, and a basic editor. Graphical Editing Framework (GEF) offers technology to create rich graphical editors and views for the Eclipse Workbench user interface. In turn, Eclipse Graphical Modeling Framework (GMF) gives a set of generative components and runtime infrastructures for developing graphical editors based on EMF and GEF.

EuGENia is a tool that automatically generates GMF models needed to implement a GMF editor from a single annotated Ecore metamodel [15] [17]. The high-level annotations provided by EuGENia simplify the complexity of GMF on generating basic eclipse-based editors and lower the entrance barrier for first steps on creating GMF editors. Due to this features, the tool is used on TerraME GIMS development to build rapid prototypes for requirements validation. It is also used to train new team members in GMF and create Eclipse based editors. Just like EuGENia, the SWTBot tool [16] was adopted to improve software development process. It is a Java based UI/functional testing tool for SWT and Eclipse based applications. Thus, the functional tests of TerraME GIMS are implemented and automated using SWTBot.

Thereby, when a new feature needs to be developed, like adding a new TerraML data type to graphical interface, a rapid prototype is developed using EuGENia and requirements are validated. Then, EMF and GMF meta-models are updated, the editor is regenerated to reflect the changes, and plug-in files (.java, .xml) are customized. New test cases are implemented using SWTBot. Finally, all the test cases are executed to ensure proper functioning of the new software release.

The Eclipse platform and its frameworks are being successfully used by several projects of similar nature [18] [19] [20] [21] [22] [23] [24] [25] [26] [27].

IV. TerraME GIMS

The TerraME GIMS (TerraME Graphical Interface for Modeling and Simulation) [28], is a visual modeling environment for developing dynamic spatiotemporal models for TerraME platform through interaction with graphical user interface components (widgets).

As mentioned above, the development of TerraME GIMS was based on the Eclipse platform due to its extensibility and availability of a wide range of public domain software frameworks. Thereby, TerraME GIMS is implemented and delivered as a set of Eclipse plug-ins, adding capabilities for building and viewing TerraME models from graphical user interface components, such as diagrams, text boxes and trees. The TerraME GIMS has a layered software architecture illustrated in Figure 3.

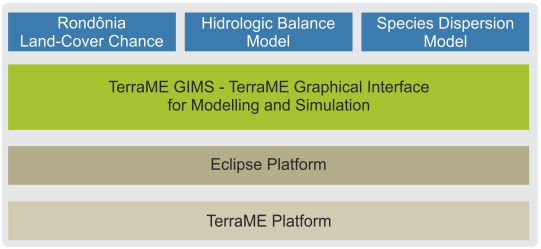

Fig. 3. TerraME GIMS architecture: a new layer between end-user and the TerraME platform

Figure 4 presents an overview of the TerraME GIMS graphical interface. Building models using TerraME GIMS is performed by Graphical Editor and by Project Explorer, Outline and Properties views. Through Project Explorer View, users may access the project files and navigate by the model hierarchically structured as a tree. Elements of the model can be edited by the diagram Editor and the Properties View. The graphical representations of the elements of the model are presented to user in the Editor and their properties can be viewed and edited from the Properties View. Furthermore, the Problems view shows errors of the model obtained from its validation. And the Console view displays the results of model execution (simulation) in TerraME platform. A Perspective was implemented to present to users this layout configuration of the Eclipse interface when working on TerraME projects.

In Editor palette, TerraML data types available for model building, such as Environment, CellularSpace, Timer, Agent, Automaton, are presented. Thus, users can compose and view the model in a diagram way (Figure 5). This facilitates the construction and identification of composition relationships between elements that constitute the model. The Editor also allows displaying and hiding the elements of the model. Besides, Outline helps users on model manipulation presenting a model overview.

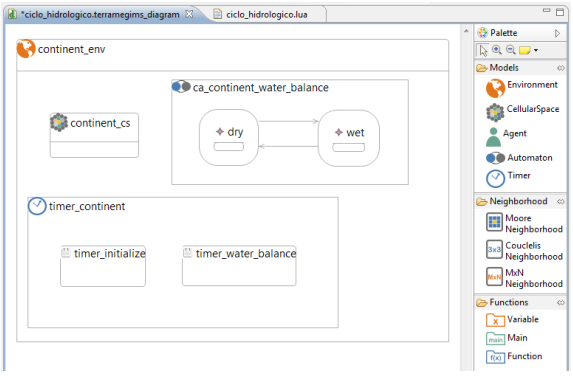

Fig. 5. TerraME GIMS Editor

Once built the model from the graphical editor and specified the properties of elements that constitute it, TerraML corresponding code can be generated and the model simulated through its execution by the TerraME interpreter.

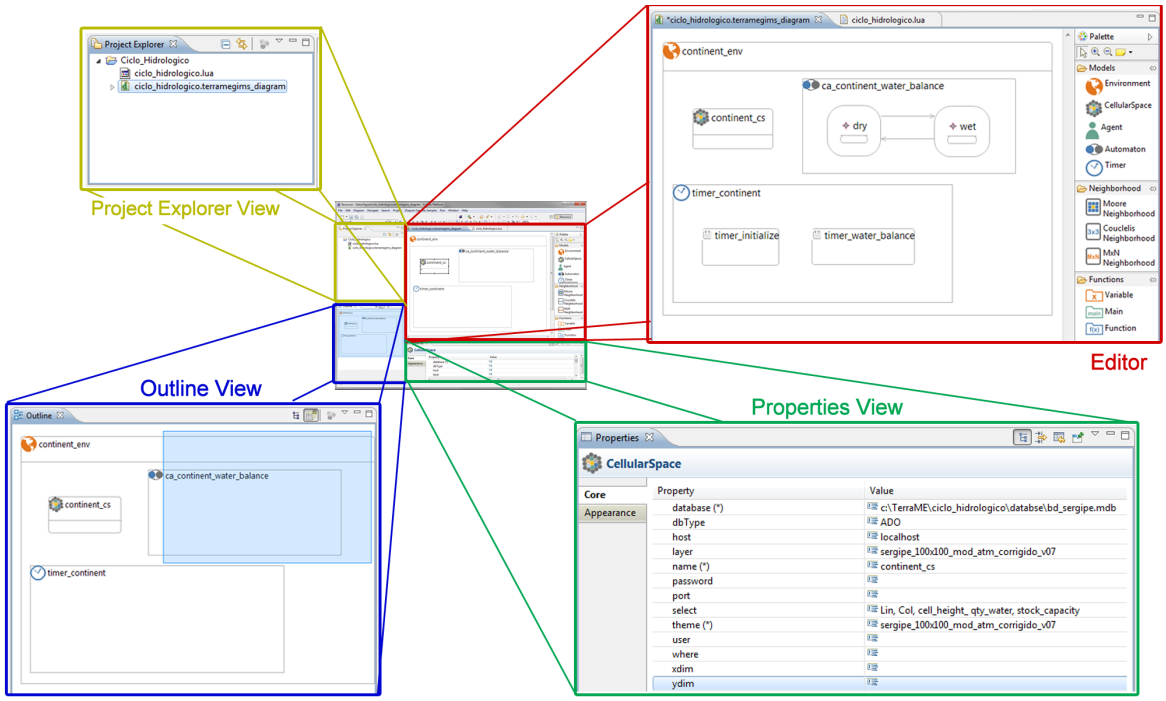

Fig. 4. Editors and Views that composes the TerraME GIMS interface.

V. Results and Discussions

TerraME GIMS was developed to provide a higher abstraction level for users, reducing their effort on syntactic aspects of the programming language. To evaluate and validate TerraME GIMS features for model building, a didactic model of the hydrological cycle was proposed [28]. This model was developed using TerraME GIMS and the corresponding TerraML source code was generated. A small part of the hydrological cycle model graphically described using TerraME GIMS and the corresponding TerraML source-code generated is presented in Figure 6.

The use of GUI for modeling is a great benefit to those users who are inexperienced in programming and adoption of practices to organize code, such as using line breaks and indentation, which contribute to readability and maintainability of it. On other hand, the standardized structure of generation code may limit freedom of users on organizing the model because of rigidity on customization of the generated code.

Moreover, the automatic code generation saves users effort on syntactic aspects of the programming language. It allows them to devote more effort on the representation (modeling) of the studied object.

The graphical representation of models also benefits the knowledge communication. The representation of the model in a higher abstraction level allows viewing and identifying its elements and relationships in a more intuitive way when compared to direct use of a programming language. Figure 7 presents a general overview of the hydrological cycle.

The current stable version of TerraME GIMS enables users to graphically create the following models (using the corresponding TerraML data types): (a) multiple scales models (Environment); (b) spatial models (CellularSpace, Neighborhood); (c) temporal models (Timer, Event and Message); (d) behavioral models (Agent, Automaton, State, Jump, Flow and Trajectory). Actually, by a project decision we will not provide creation and edition of Cells using the graphical interface. GIS tools are better for this purpose. About features as creating variables and functions, it is also possible to create them using GUI. Although it is possible to describe the execution flow through flowcharts, another decision was not to offer this feature for now. Thereby, user only has to insert commands on the model as control structures and mathematical operations. And all the TerraML source code will be generated by TerraME GIMS.

A. Lessons Learned

Using the Eclipse platform and its frameworks made the development and maintenance of TerraME GIMS software more efficient. Changes in TerraML language could be rapidly incorporated by the GUI and by the code generator. Nevertheless, keeping TerraME GIMS constantly updated with Eclipse platform is challenging due to incompatibility between versions of frameworks and tools used in development. For instance, SWTBot testing tool is incompatible with Juno version of Eclipse and, therefore, automated tests are still being conducted only in Helios version. However, automation of functional testing is necessary for a good productivity and quality assurance. Thus, when adopting the development based on plug-ins, it is critical to consider the risks coming from the dependence on the technologies used and to evaluate the feasibility of following the updates of the base platform.

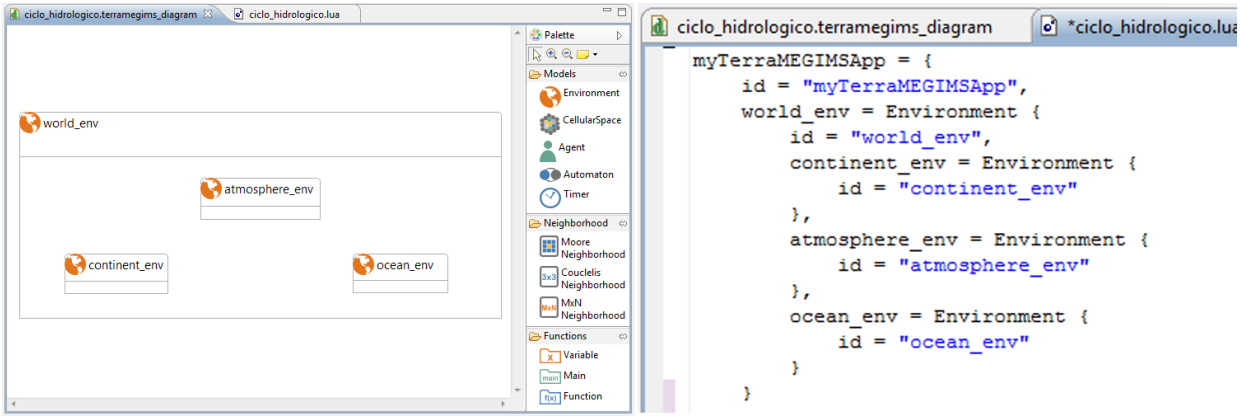

Fig. 6. Definition of nested environments using TerraME GIMS: (a) graphical representation; (b) corresponding TerraML source code.

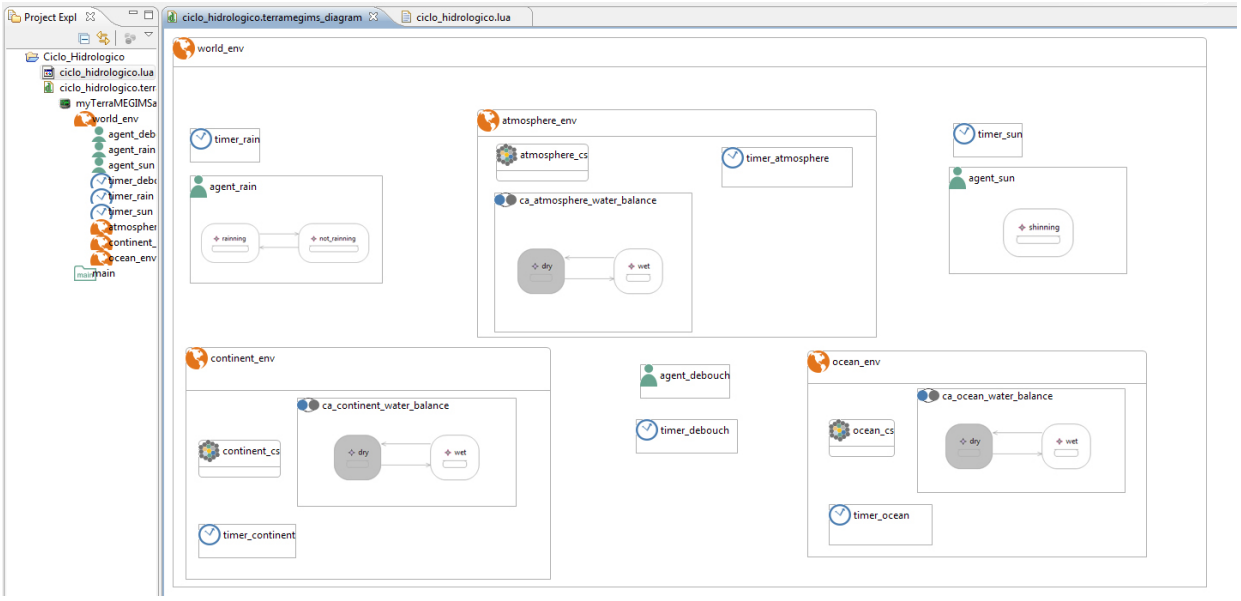

Fig. 7. General overview of the hydrological cycle model.

The facilities offered by Eclipse for building tools based on its IDE and the use of frameworks that support the model-driven development (MDD) has enabled rapid prototyping of the first versions of TerraME GIMS. The initial training of new developers was facilitated by the use of mature frameworks like EMF, GEF and GMF. However, building features that require in-depth knowledge is hampered by literature that consists in its majority only of superficial examples.

One of the main advantages of development based on plug-ins is the integration with third party tools, adding important values for the user with minimum cost. For instance, integration with LuaEditor plug-in provided highlighting syntax features. Thus, part of the development effort consists in researching and analyzing other plug-ins that are available. Relevant points in this analysis are: maturity, maintenance and development of plug-ins; size of the user community; quality of available documentation.

Another point that has led us to development based on plug-in over Eclipse platform was thinking of a long term strategy for TerraME GIMS software. We hope that the adoption of an open source platform with a large community of developers and users may ensure the future of TerraME GIMS. The most important challenge will be to encourage contributions and improvements made by users and developers, making the software sustainable. Therefore, despite the difficulties encountered throughout development, we believe development based on plug-ins to be a good long-term strategy.

VI. Conclusions

There is a growing demand for environmental models to help researchers, entrepreneurs and governments in the process of decision making aiming for interventions with smaller impacts to environment. Modeling and simulation favor the understanding of the interaction between biophysical and human processes. However, building environmental models

is a complex task, which requires the use of computational tools and involvement of a multidisciplinary team. This team is formed by experts in the domain problem but, generally, they do not have solid knowledge about algorithms and programming techniques.

Thus, to make effective the use of all services offered by TerraME platform, it is essential to offer users a higher level of abstraction for building models. So, a graphical interface for development models is been developed. This tool denominated TerraME GIMS (TerraME Graphical Interface for Modeling and Simulation) offers a graphical interface that enables its users to build dynamic spatial models for TerraME platform.

The usage of graphical interface supported by TerraME GIMS makes it easier to describe conceptual models of studied phenomena, enables identifying the model components and the relationships between them in more intuitive way, and reduces efforts on syntactic aspects of TerraML programming language.

The strategy of using a plug-in based development and Eclipse platform and frameworks allows faster results and consequently, the realization of this work. Now, as future works, we intend to evaluate TerraME GIMS with multidisciplinary users group.

ACKNOWLEDGMENT

We kindly acknowledge the fruitful discussions with the TerraME Modeling Group. This work was partially funded by CNPq (CTINFO 09/2010), FAPEMIG and UFOP.

REFERENCES

[1] M. G. Turner, R. H. Gardner, and R. V. OŃeill, *Landscape Ecology in Theory and Practice: pattern and process*. New York: Springer-VerLag, 2001.

[2] P. Bratley, B. L. Fox, and L. E. Schrage, *A guide to Simulation*, 2nd ed. Springer, 1987.

[3] B. M. Pedrosa and G. Câmara, *Modelagem dinâmica e sistemas de informações geográficas*, 1st ed. Embrapa, 2007, ch. 5, pp. 235–280, in Meirelles, M.S.P.; Câmara, G.; Almeida, C.M. (org.): Geomática - Modelos e aplicações ambientais.

[4] L. von Bertalanffy, *General system theory*. Braziller, 1968.

[5] M. Wooldridge, N. R. Jennings *et al.*, "Intelligent agents: Theory and practice," *Knowledge engineering review*, vol. 10, no. 2, pp. 115–152, 1995.

[6] A. W. Burks and J. Von Neumann, *Theory of self-reproducing automata*. Urbana: University of Illinois Press, 1966.

[7] B. P. Zeigler, H. Praehofer, and T. G. Kim, *Theory of modeling and simulation: Integrating discrete event and continuous complex dynamic systems*. Academic Pr, 2000.

[8] T. G. S. Carneiro, "Nested-ca: a foundation for multiscale modeling of land use and land change," Ph.D. dissertation, Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, Brasil, 2006.

[9] T. G. S. Carneiro, T. F. M. de Lima, and S. D. Faria, "Terralab - using free software for earth system research and free software development," in *Proceedings of the X Workshop of Free Software*. Porto Alegre, RS, Brasil: Sociedade Brasileira de Computação (SBC), 2009, pp. 35–40.

[10] P. R. de Andrade, A. M. V. Monteiro, G. Câmara, and S. Sandri, "Games on cellular spaces: How mobility affects equilibrium," *Journal of Artificial Societies and Social Simulation*, vol. 12, no. 1, p. 5, 2009. [Online]. Available: http://jasss.soc.surrey.ac.uk/12/1/5.html

[11] E. G. Moreira, A. P. D. de Aguiar, S. S. Costa, and G. Câmara, "Spatial relations across scales in land change models," in *Proceedings of the X Brazilian Symposium on GeoInformatics*. Rio de Janeiro, RJ, Brasil: Sociedade Brasileira de Computação (SBC), 2008, pp. 95–107.

[12] P. F. Pimenta, A. Coelho, S. S. Costa, E. G. Moreira, A. P. Aguiar, G. Câmara, R. Araújo, and A. Ribeiro, "Land change modeling and institutional factors: heterogeneous rules of territory use in the brazilian amazonia." in *Proceedings of the X Brazilian Symposium on GeoInformatics*. Rio de Janeiro, RJ, Brasil: Sociedade Brasileira de Computação (SBC), 2008, pp. 81–93.

[13] C. C. Gibson, E. Ostrom, and T. K. Ahn, "The concept of scale and the human dimensions of global change: a survey," *Ecological Economics*, vol. 32, no. 2, pp. 217–239, 2000. [Online]. Available: http://econpapers.repec.org/RePEc:eee:ecolec:v:32:y:2000:i:2:p:217-239

[14] Eclipse. Eclipse documentation. [Online]. Available: http://help.eclipse.org/

[15] ——. Eugenia website. [Online]. Available: http://www.eclipse.org/epsilon/doc/eugenia/

[16] ——. Swtbot website. [Online]. Available: http://www.eclipse.org/swtbot/

[17] D. Kolovos, L. Rose, S. Abid, R. Paige, F. Polack, and G. Botterweck, "Taming emf and gmf using model transformation," in *Model Driven Engineering Languages and Systems*, ser. Lecture Notes in Computer Science, B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, and B. Apduhan, Eds. Springer Berlin / Heidelberg, 2010, vol. 6394, pp. 211–225.

[18] M. Fontana, F. F. Costa, U. B. Sangiorgi, P. A. D. Bichara, D. G. aes, and A. C. de C. Lima, "Hiperioncad: a cad tool for design and optimization of optical telecommunication," in *10th International Conference on Advanced Communication Technology - ICACT 2008*, vol. 3, 2008, pp. 2027–2032.

[19] D. A. Sadilek and G. Wachsmuth, "Prototyping Visual Interpreters and Debuggers for Domain-Specific Modelling Languages," in *4th European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'08)*, ser. Lecture Notes in Computer Science, I. Schieferdecker and A. Hartman, Eds., vol. 5095. Springer-Verlag, 2008, pp. 63–78.

[20] K. Ehrig, C. Ermel, S. Hänsgen, and G. Taentzer, "Generation of visual editors as eclipse plug-ins," in *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering*, ser. ASE '05. New York, NY, USA: ACM, 2005, pp. 134–143. [Online]. Available: http://doi.acm.org/10.1145/1101908.1101930

[21] K. Krogmann and S. Becker, "A case study on model-driven and conventional software development: The palladio editor," *Software Engineering*, pp. 169–176, 2007.

[22] T. Buchmann, A. Dotor, and B. Westfechtel, "Model-driven development of graphical tools - fujaba meets gmf." in *ICSOFT (SE)*, J. Filipe, B. Shishkov, and M. Helfert, Eds. INSTICC Press, 2007, pp. 425–430.

[23] M. Lipaczewski, S. Struck, and F. Ortmeier, "SAML goes Eclipse - Combining Model-Based Safety Analysis and High-Level Editor Support," in *Proceedings of the 2nd International Workshop on Developing Tools as Plug-Ins (TOPI)*. IEEE, 2012, pp. 67–72.

[24] S. Zachariadis and T. Cianchi, "Architecting a plug-in based steam turbine design tool," in *Proceedings of the 1st Workshop on Developing Tools as Plug-ins*, ser. TOPI '11. New York, NY, USA: ACM, 2011, pp. 57–57. [Online]. Available: http://doi.acm.org/10.1145/1984708.1984726

[25] P. E. Salas, E. Marx, A. Mera, and J. Viterbo, "Rdb2rdf plugin: relational databases to rdf plugin for eclipse," in *Proceedings of the 1st Workshop on Developing Tools as Plug-ins*, ser. TOPI '11. New York, NY, USA: ACM, 2011, pp. 28–31. [Online]. Available: http://doi.acm.org/10.1145/1984708.1984717

[26] G. de Caso, D. Garbervetsky, and D. Gorín, "Pest: from the lab to the classroom," in *Proceedings of the 1st Workshop on Developing Tools as Plug-ins*, ser. TOPI '11. New York, NY, USA: ACM, 2011, pp. 5–8. [Online]. Available: http://doi.acm.org/10.1145/1984708.1984711

[27] V. Chiprianov, Y. Kermarrec, and S. Rouvrais, "On the extensibility of plug-ins," in *ICSEA: 6th International Conference on Software Engineering Advances*, Barcelona, Espagne, 2011, pp. 557–562, 11003 11003. [Online]. Available: http://hal.archives-ouvertes.fr/hal-00725543

[28] T. F. M. de Lima, S. D. Faria, and T. G. de Senna Carneiro, "Development of a didactic model of the hydrologic cycle using the terrame graphical interface for modeling and simulation," in *Computational Science and Its Applications - ICCSA 2011*, ser. Lecture Notes in Computer Science, B. Murgante, O. Gervasi, A. Iglesias, D. Taniar, and B. Apduhan, Eds. Springer Berlin / Heidelberg, 2011, vol. 6785, pp. 75–90. [Online]. Available: http://dl.acm.org/citation.cfm?id=2029365.2029373