A Domain Specific Language for Spatial Simulation Scenarios(DSL3S): Introduction and Tool Support

Luís de Sousa
CRP Henri Tudor
6A, Av. des Hauts-Fourneaux
L-4362 Esch-sur-Alzette
Luxembourg
luis.a.de.sousa@gmail.com

Alberto Rodrigues da Silva
IST / INESC-ID
Rua Alves Redol, 9
1000-029 Lisboa
Portugal
alberto.silva@tecnico.ulisboa.pt

ABSTRACT

Cellular automata and agent-based modelling techniques have long been used for spatial simulation in the Geographic Information Systems field. However, they largely rely on code libraries and pre-compiled models, either requiring advanced programming skills or imposing scope constraints. Several domain specific languages have been proposed in this context, but mostly resulting in new textual programming languages.

DSL3S is a domain specific language for spatial simulation, synthesising concepts in a UML profile, permitting the design of simulation models through graphical elements. MDD3S is an implementation of this language relying on model-driven development (MDD) tools built around the Eclipse IDE; it produces ready to run simulations from DSL3S models, supported by the MASON simulation tool-kit. These assets have proved sufficient to developed classic models in different GIS application fields.

Categories and Subject Descriptors

CM [**Coordination Models**]: Languages and Applications

Keywords

Domain Specific Language, Spatial Simulation, UML Profile, Model-Driven Development

1. INTRODUCTION

In the Geographic Information Systems (GIS) domain, exploring how spatial variables and features evolve with time is often necessary. To this purpose several techniques have been developed, comprising a sub-domain of GIS referred as Spatial Simulation [1]. Throughout the past two decades various code libraries and tools have been made available to researchers and analysts in this field. However, they still pose important challenges, starting with a non trivial choice for the most suitable tool, plus the requirement for solid programming skills or in exchange the compromise of application scope.

Among the many Spatial Simulation tools available today two essential groups stand out: *Program-level* and *Model-level* tools [2]. The first are conceived for the programmer, mostly code libraries that encapsulate some of the complexity in specific methods or functions. Examples are REPast [12] and MASON [7]. By their very nature, *Program-level* tools are not accessible to spatial analysts lacking programming skills and may require a long learning process. In contrast, *Model-level* tools, such as LANDIS [9] or TELSA [8], are pre-programmed models that can parametrised, setting inputs and tuning pre-defined variables. They are easier to use, but also restrict the application scope; in some cases integration with spatial data is poor or non-existent.

Beyond these difficulties it has been recognised that an integrated approach to the description of agent-based models is largely lacking [11]. The reliance on source code or static documentation can create extra barriers when communicating model dynamics to stakeholders or peer analysts; model comparison and reuse are also difficult.

The *Domain Specific Language for Spatial Simulation Scenarios* (DSL3S) is a Domain Specific Language (DSL) that tries to ease the development of spatial simulations through a Model-Driven Development (MDD) [13] approach. It proposes a development processes through the arrangement of graphical elements and their relationships, dispensing formal programming knowledge. These graphical models can then be translated into ready to run simulations through the application of a code generation infrastructure.

The MDD approach raises the level of abstraction at which development takes place, thus simplifying the communication between analysts and stakeholders [10]. It can also allow prototyping by non-programmers. By detaching model development from specific technologies, it can improve interoperability with geo-spatial data, generating *ad hoc* code as needed. Lastly, it can lay the foundations for a standard language in this domain, as successful efforts in parallel fields have proved, such as SysML[1] (for systems engineering) or ModelicaML[2] (for complex systems).

DSL3S relies on the MDD standard issued by the Object Management Group (OMG): Model-Driven Architecture (MDA)[3], that promotes UML profiles for the definition of DSLs. UML 2.0 allows the extension of its core primi-

SAC'15 April 13-17, 2015, Salamanca, Spain.
Copyright 2015 ACM 978-1-4503-3196-8/15/04...$15.00.
http://dx.doi.org/10.1145/2695664.2695972

[1] http://www.sysml.org/

[2] https://www.openmodelica.org/index.php/home/tools/134

[3] http://www.omg.org/mda/

tives (graphical elements, links, etc) through specialisation for different application domains.

2. RELATED WORK

There have been several attempts to create DSLs for Spatial Simulation. They present ways to bridge the gap between *Program-level* and *Model-level* tools, approaching model description to natural language, but still retaining some of the freedom of general purpose programming languages.

NetLogo is a specialisation of the Logo functional programming language, that started as an educational project directed at Agent based simulations. It is relatively easy to learn, free of advanced programming concepts.

SELES (Spatially Explicit Landscape Event Simulator) is a declarative DSL for Landscape Dynamics [5]. It supports a vast range of different input raster formats (most common in Land Use / Land Cover data).

MOBIDYC (Modelling Based on Individuals for the Dynamics of Communities) is a Smalltalk code package for the study of population dynamics [6], defining a set of simple primitives designed to approach the language to natural language. However, it has no support for the direct input of geo-spatial data.

Ocelet is a declarative DSL for landscape dynamics, employing the concept of service-oriented architecture [4]. Models are built by components interacting with each other through services.

These DSLs focus mainly on providing a refined concrete syntax but still framed in older programming paradigms such as declarative or functional languages. They require the understanding of keywords and how to compose a coherent set of instructions or declarations into a specific model. Lack of interoperability with geo-spatial data is also an issue in some cases, as so platform dependency. In essence these efforts relying on textual languages fall into the same pitfalls identified by [14] regarding fourth generation languages: they struggle to hike the level of abstraction at which model development takes place.

3. DSL3S: LANGUAGE AND TOOLS

This section provides a brief description of DSL3S and the prototype framework developed to support it. A more detailed account of the language can be found in [3].

3.1 Language

DSL3S identifies two main constructs composing a spatial simulation: Animats and Variables. **Animat** signifies *artificial animal*, in DSL3S standing for all spatial elements that either evolve themselves or induce change in their surroundings; examples are: fire (in a wildfire model) or predators (in a population dynamics model). **Spatial** variables are spatial information layers that have some sort of impact on the dynamics of a simulation, e.g. slope that deters urban sprawl or biomass that feeds a wildfire. **Global** variables provide information that is constant across the space of simulation, such as capital stock in an urban sprawl model.

An Animat is composed by a set of **Attributes**, that describe its internal state at each moment in time. **Operations** make explicit the way animats act and react to the environment, thus encoding spatial dynamics. Animat operations considered in DSL3S are:

- **Emerge**: sets the conditions under which a new animat can appear in the simulation space. An example may be an urban sprawl simulation where the emergence of new urban spots is determined by the distance to transport infrastructure or topography.

- **Move**: relates an animat with spatial variables or with other animats, determining the locations that are more or less favourable to be in.

- **Replicate**: captures behaviours where an animat replicates itself, such as an organism reproducing in a biological simulation.

- **Supply**: provides access to the animat internal properties, thus supplying resources or information to other animats.

- **Harvest**: determines which assets an animat seeks: resources or information provided by other animats or variables.

- **Perish**: defines the circumstances under which an animat ceases to exist; a fire extinguishing is an example.

DSL3S restricts its set of operations for three reasons: (i) to keep the language compact and easy to learn; (ii) more refined operations are less common in spatial simulation applications and may eventually be composed with simpler primitives; and (iii) to insulate the user from the technical implementation details in the choice between Cellular Automata and Agent based models.

Figure 1 presents these key constructs in a conceptual model. Each of these constructs is realised by a specific stereotype in the DSL3S UML profile; a stereotype exists also for each operation type. A Simulation is composed by a set of Animats, Spatial and Global variables; Animats are composed by a set of Attributes and Operations, that determine how their internal state evolves. An animat acts through different types of Operations, that can induce changes on spatial variables or the state of other animats.

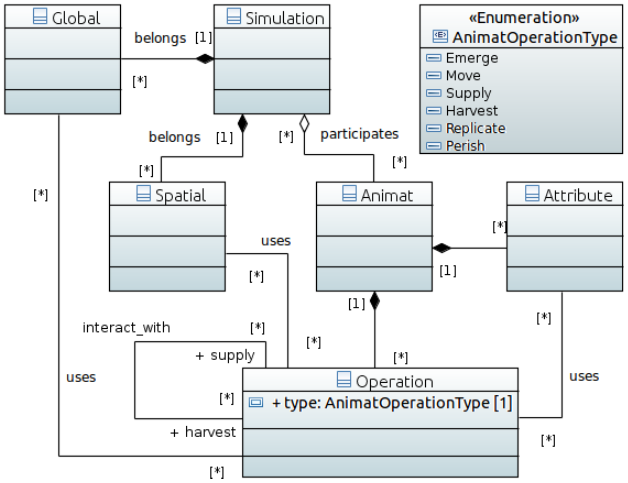

Figure 1: The DSL3S meta-model.

3.2 Tool Support

Model Driven Development for Spatial Simulation Scenarios (MDD3S) is the name of the prototype framework developed to support the DSL3S language. MDD3S relies solely on open source tools: (i) **Papyrus** - an Eclipse[4] add-on for UML modelling; (ii) **Acceleo** - another Eclipse add-on supporting model-to-code generation templates; (iii) **MASON** - a *Program-level* spatial simulation framework used as a library by the code generated.

Papyrus is a graphical editor for the UML language based on the Eclipse Modelling Framework[5] (EMF). Papyrus evolved to support the development of *ad hoc* DSLs, through the definition of UML profiles.

Acceleo[6] is an open source code generator also built on EMF. Acceleo interprets templates written with the MOF Model to Text Transformation Language[7] (MOFM2T), also an OMG standard. It fully supports code generation from meta-models, identifying stereotypes applied on classes and providing access to its properties.

MASON (acronym for "Multi-Agent Simulator Of Neighbourhoods") aims to be a light-weight, highly portable, multipurpose agent-based modelling package [7]. It is fully written in Java and open source. GeoMASON[8] is an extension that provides Java objects to deal specifically with georeferenced data, providing input and output functionality with various raster and vector data formats.

4. DISCUSSION AND FUTURE WORK

The DSL3S UML profile are the MDD3S framework are in the public domain[9], and may be installed as plug-ins to Eclipse. These assets are able to translate a graphical, abstract and platform independent model produced with DSL3S into a coded simulation.

Several case studies are also available that can be accessed with Papyrus or any other software able to interpret the XMI language. These case studies showcase the employment of DSL3S in different applications traditionally targeted by spatial simulation, such as population dynamics, widfires or urban sprawl. At this moment DSL3S and MDD3S are at least capable of prototyping.

DSL3S will be further assessed through its application to real world scenarios. An iterative process shall provide an understanding of how far it can go in its current form and if extensions are necessary. Graphical semantics is another area where improvements are possible, in particular through the employment of stereotype icons proposed before [3]. Presently, such feature is not fully supported by the MDD tools used.

In the near future a series of support contents will be released to facilitate the first contact with the language. With these contents an evaluation experience will be launched where international spatial simulation experts will be invited to test the language and feed back on its usability and productivity.

[4] http://www.eclipse.org/modeling

[5] http://www.eclipse.org/modeling/emf/

[6] http://www.acceleo.org/pages/introduction/en

[7] http://www.omg.org/spec/MOFM2T/1.0/

[8] http://cs.gmu.edu/ eclab/projects/mason/extensions/geomason/

[9] https://github.com/ldesousa/DSL3S

5. REFERENCES

[1] M. J. de Smith, M. F. Goodchild, and P. A. Longley. *Geospatial Analysis: A Comprehensive Guide to Principles, Techniques and Software Tools - Fourth Edition*, chapter Geosimulation, pages 625–672. Winchelsea Press, 2013.

[2] L. de Sousa and A. R. da Silva. Review of spatial simulation tools for geographic information systems. In *SIMUL 2011*. ThinkMind, 2011.

[3] L. de Sousa and A. R. da Silva. Preliminary design and implementation of dsl3s - a domain specific language for spatial simulation scenarios. In *Proceedings of CAMUSS - Cellular Automata Modelling for Urban and Spatial Systems*, pages 269–280. Dep. of Civil Engineering of the Univ. of Coimbra, 2012.

[4] P. Degenne, D. Lo Seen, D. Parigot, R. Forax, A. Tran, A. Ait Lahcen, O. Curé, and R. Jeansoulin. Design of a Domain Specific Language for modelling processes in landscapes. *Ecological Modelling*, 220:3527–3535, 2009.

[5] A. Fall and J. Fall. A domain-specific language for models of landscape dynamics. *Ecological Modelling*, 141:1–18, 2001.

[6] V. Ginot, C. Le Page, and S. Souissi. A multi-agents architecture to enhance end-user individual based modelling. *Ecological Modelling*, 157:23–41, 2002.

[7] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan. Mason: A multi-agent simulation environment. *Simulation: Transactions of the society for Modeling and Simulation International*, 82(7):517–527, 2005.

[8] J. Merzenich and L. Frid. Projecting Landscape Conditions in Southern Utah Using VDDT. In M. Bevers and T. M. Barrett, editors, *Systems Analysis in Forest Resources: Proceedings of the 2003 Symposium*, pages 157–163, Portland, OR, 2005.

[9] D. Mladenoff. LANDIS and forest landscape models. *Ecological Modelling*, 180(1):7 – 19, 2004.

[10] P. Mohagheghi, W. Gilani, A. Stefanescu, M. Fernandez, B. Nordmoen, and M. Fritzsche. Where does model-driven engineering help? Experiences from three industrial cases. *Software & Systems Modeling*, 12(3):619–639, 2013.

[11] B. Müller, S. Balbi, C. M. Buchmann, L. de Sousa, G. Dressler, J. Groeneveld, C. J. Klassert, Q. B. Le, J. D. Millington, H. Nolzen, D. C. Parker, J. G. Polhill, M. Schlüter, J. Schulze, N. Schwarz, Z. Sun, P. Taillandier, and H. Weise. Standardised and transparent model descriptions for agent-based models: Current status and prospects. *Environmental Modelling & Software*, 55(0):156 – 163, 2014.

[12] M. J. North, T. R. Howe, N. T. Collier, and J. R. Vos. The Repast Simphony development environment. In *Proceedings of the Agent 2005 Conference on Generative Social Processes, Models, and Mechanisms*, Chicago, USA, 2005.

[13] B. Selic. The Pragmatics of Model-Driven Development. *IEEE Software*, 20(5):19–25, 2003.

[14] B. Selic. Personal reflections on automation, programming culture, and model-based software engineering. *Automated Software Engineering*, 15(3-4):379–391, 2008.