COMPUTER SCIENCE | RESEARCH ARTICLE

Usability evaluation of the domain specific language for spatial simulation scenarios

Luís Moreira de Sousa1* and Alberto Rodrigues da Silva2

Abstract: This article presents the results of a usability evaluation initiative conducted on the Domain Specific Language for Spatial Simulation Scenarios (short name DSL3S) and its supporting tools. This language applies a Model-Driven Development approach to spatial simulation, providing model development through the composition of graphical elements and the subsequent transformation to source code. Potential users trained in disciplines related to Geographic Information Systems were exposed for the first time to the language with an introductory exercise. After installing the supporting tools and developing a simple spatial simulation model, participants then evaluated the language and its tools by answering a questionnaire. The results of this evaluation point to a good degree of usability, with particularly positive appreciations of the DSL3S supporting tools. Notwithstanding, participants also show some reluctance in adopting such a development framework, hinting at some reminiscent scepticism towards domain specific modelling languages and Model-Driven Development.

Subjects: Simulation & Modeling; Computer Science (General); Geographic Information Systems

Keywords: spatial simulation; domain specific modelling language; UML profile; model-driven development; model driven architecture (MDA)

ABOUT THE AUTHORS

Luís Moreira de Sousa is a PhD (2016) in Computer Science and a MSc in both Geographic Information Systems (2005) Computer Science (2002). He has conducted research in multiple institutions across Europe, including the University of Lisbon, the Luxembourg Institute of Science and Technology (LIST), the Swiss Federal Institute of Aquatic Science and Research (EAWAG) and presently at ISRIC – World Soil Information. Spatial Simulation, Model-Driven Development and Geo-computation in the Geodetic domain are his main interests.

Alberto Rodrigues da Silva is a PhD (1999) and a MSc (1992) in Computer Science. He is an associate professor at the Computer and Information Systems Department of the University of Lisbon (IST/UL). He is author of five technical books and more than 150 peer-reviewed articles published in scientific journals, conferences and workshops proceedings. He has research interests, among others, in Modelling and Meta-modelling, Model-Driven Engineering, Requirements Engineering.

PUBLIC INTEREST STATEMENT

Model-Driven Development (MDD) is an approach to software development that centres around graphical models, casting source code to the background. As it largely dispenses coding, this approach has been used to create Domain Specific Modelling Languages (DSMLs) that provide expressive and understandable means for software engineers to involve stakeholders in the development process. While many of such languages have been developed, no thorough methodology has emerged to evaluate their effectiveness; in fact most DSMLs are never evaluated.

This article reports an evaluation conducted on DSL3S – a DSML for spatial simulation. Potential users with professional backgrounds close to the domain were exposed for a first time to DSL3S and its supporting tools, conducting a small exercise and then providing feedback through a questionnaire. The results are not straightforward, while participants rate positively DSL3S and its tools, a certain degree of scepticism towards MDD remains.
1. Introduction

There have been several attempts to create Domain Specific Languages (DSLs) for Spatial Simulation, with the aim of bridging the gap between code libraries and pre-programmed models (de Sousa & Silva, 2011). Ocelet (Degenne et al., 2009), GAML (Grignard et al., 2013) or NetLogo (Berryman, 2008) are good examples of these tries, each with different degrees of success. These languages bring model description closer to natural language, but still retain some of the development freedom provided by general purpose programming languages. These previous efforts focus on providing a refined concrete syntax but remain framed in old programming paradigms, invariably producing textual languages. They require the understanding of keywords and the composition of a coherent set of instructions or declarations into a specific model. Some of these tools present interoperability issues with geo-spatial data, as so platform dependencies. In essence these efforts fall into the same pitfalls identified by Selic (2008) regarding fourth generation languages: they struggle to hike the level of abstraction at which model development takes place.

The Domain Specific Language for Spatial Simulation Scenarios (DSL3S) (de Sousa & Silva, 2016) is a domain specific modelling language (DSML) that intends to ease the development of spatial simulations through a Model-Driven Development (MDD) (Selic, 2003) approach. MDD emerged out of the Model-Driven Engineering field (Silva, 2015) in Computer Science. If initially focused on improving software development and documentation, this methodology yields important potentials that go beyond. By raising the level of development abstraction, the MDD approach simplifies the communication between model developers and stakeholders (Mohagheghi et al., 2013). It can also allow prototyping by non-programmers, since development becomes a graphical activity. Furthermore, 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, as successful efforts in parallel fields have proved, such as SysML (http://www.sysml.org/) (for systems engineering) or ModelicaML (https://www.openmodelica.org/index.php/home/tools/134) (for complex systems).

Following this approach, DSL3S is realised as a UML profile, in which key concepts of spatial simulation (e.g. spatial layer, agent, behaviour, state) are captured as UML stereotypes. With this UML profile graphical models specific to the spatial domain can be developed and tuned, using the properties invested by each stereotype. These graphical models can then be translated into ready to run simulations through the application of a model-to-code transformation infrastructure. Therefore, the development processes of spatial simulation models with DSL3S and its accompanying tools dispenses formal programming knowledge. Whereas DSL3S has already shown sufficient to tackle traditional and diverse spatial simulation scenarios (de Sousa & Silva, 2015), before the experiment reported herein, it was unknown what this language precisely adduces to potential users, spatial analysts and practitioners of related disciplines.

Challenger, Kardas, and Tekinerdogan (2015) note that DSMLs are generally developed without formally undergoing an evaluation programme, in great measure since a thorough DSML evaluation framework is yet to be established. These authors propose a broad framework for the evaluation of DSMLs developed for multi-agent systems, in which questionnaires are included as a means of qualitative end user evaluation. Morais and Silva (2015) proposed an evaluation framework quantifying the quality of user interface DSMLs, but noting that usability and tool support are not straightforwardly quantifiable aspects, since they can be unique to each DSML. In this vein, Ewais and De Troyer (2014) conducted a usability study on a DSML for Virtual Learning Environments through a language introduction exercise, followed by a qualitative questionnaire.

The need to understand the usability and user receptivity of DSL3S thus lead to the evaluation initiative presented in this article. This initiative followed along the methodological lines applied by Ewais and De Troyer (2014). Throughout various sessions, professionals in Geographic Information Systems (GIS), computer science and related fields participated in a development exercise, afterwards providing their impressions through a questionnaire. This evaluation meant to understand
how potential users fare in a first contact with DSL3S and how they regard the approach: (i) assess how easy or hard is to install and use the supporting tools; (ii) assess if users unfamiliar with DSL3S can effectively develop prototype models with it; and (iii) obtain feedback on the potential adoption of the language and the MDD approach in general. The results not only shed some light on DSL3S itself, they also carry important clues on the perception professionals of fields diverse to Computer Science have on MDD and DSMLs.

This article is organised as follows: Section 2 briefly describes the DSL3S language; Section 3 presents the tools supporting DSL3S and the software infrastructure underpinning them; Section 4 describes the exercise undertaken by participants in the evaluation sessions and the profiles of the participants; Section 5 details the results of the questionnaire; Section 6 interprets these results and Section 7 offers some concluding remarks.

2. Language

DSL3S is built around three basic concepts: Spatial variables, Animats and Operations. 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. Animat is a term coined by Wilson (1991), signifying artificial animal, representing all spatial elements that evolve themselves or induce change in their surroundings; e.g. fire in a wildfire model, urban areas in an urban sprawl model.

Beyond these core concepts, other elements can also be found in a simulation, such as Global variables, setting information that is constant across the space of simulation, like capital stock in an urban sprawl model. A set of Attributes is required to compose an Animat, describing the internal state of each instance. Operations capture the way animats act and react to the environment, i.e. they encode the spatial dynamics. Six basic types of Animat operations are considered in DSL3S:

- **Emerge**: determines the conditions under which a new instance of an animat can appear in the simulation space; an example may be an urban development simulation where the emergence of new urban spots is possible in an area that meets a certain set of criteria.
- **Move**: relates an animat with spatial variables or with other animats, determining the locations that are more or less favourable to be in.
- **Replicate**: sets the conditions for an animat to replicate itself, such as an organism in a biological simulation reproducing a sibling.
- **Supply**: provides access to internal properties of animats, thus supplying resources or information to other animats.
- **Harvest**: an act by which an animat may change other elements in its surroundings; it can act on a spatial variable, such as a wildfire consuming bush, or by seizing resources from another animat.
- **Perish**: defines the circumstances under which an animat may cease to exist during simulation; examples can be a biological animal starving or a fire extinguishing.

This strict set of operations tries to match the essential properties of an agent, as outlined by Franklin and Grasser (1997)—autonomous, continuous, reactive, proactive and mobile—with the core concepts found in Cellular Automata—state, neighbourhood, transition rules and time (Ermentrout & Edelstein-Keshet, 1993).

Figure 1 presents these key constructs in a conceptual model. A Simulation is composed by a set of Spatial variables, Animats and Global variables; Animats are composed by a set of Attributes and Operations, the latter determining how their internal state evolves. An animat acts through different types of Operations, that can induce changes on global variables, spatial environment or the state of other animats. Full details of the language can be found in de Sousa and Silva (2016).
DSL3S relies on the MDD standard issued by the Object Management Group (OMG)—Model-Driven Architecture (MDA) (http://www.omg.org/mda)—that promotes UML profiles for the definition of DSMLs. UML 2.0 allows the extension of its core primitives (graphical elements, links, etc) through specialisation for different application domains. DSL3S is therefore formalised as a UML profile, with each of the language constructs identified above realised as a UML stereotype.

3. Tools
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: Papyrus, Acceleo and MASON. The full software stack is depicted in Figure 2.

Papyrus is a graphical editor for the UML language based on the Eclipse Modelling Framework (http://www.eclipse.org/modeling/emf/) (EMF). It allows the edition and visualisation of structured models defined with the XMI language. Papyrus evolved to support the development of ad hoc DSMLs, through the definition of UML profiles. Presently it supports version 2 of the UML language in almost its entirety.

Acceleo (http://www.acceleo.org/pages/introduction/en) is an open source model-to-code transformation framework also built on EMF. Acceleo interprets templates written with the MOF Model to Text Transformation Language (http://www.omg.org/spec/MOFM2T/1.0/) (MOFM2T), also an OMG standard. It fully supports code generation from meta-models, identifying stereotypes applied on classes and providing access to its properties.

The Multi-Agent Simulator Of Neighbourhoods (MASON) is a light-weight, highly portable, multi-purpose agent-based modelling package written in Java (Luke, Cioffi-Revilla, Panait, Sullivan, &
MASON is open source and likely the fastest of the most popular spatial simulation tool-kits (Railsback, Steven, & Jackson, 2006). GeoMASON (http://cs.gmu.edu/eclab/projects/mason/extensions/geomason/) is an extension that provides Java objects to deal specifically with geo-referenced data.

The DSL3S UML profile and its accompanying MDD3S framework are publicly available at the code sharing platform GitHub (https://github.com/MDDLingo/DSL3S). Various assets were developed to facilitate their installation, apprenticeship and usage. They include: (i) two plug-ins for Eclipse, (ii) a web-based manual, (iii) a series of tutorial videos and (iv) a set of example models.

Two different Eclipse plug-ins were developed, one for the DSL3S UML profile and another for MDD3S. The profile plug-in registers DSL3S with the Eclipse platform, making the language available to any Papyrus model developed in the platform. The user is thus able to apply the profile on models, gaining access to all its stereotypes and respective properties.

The MDD3S plug-in registers a number of JAR files with Eclipse that execute the model-to-code transformation and creates an additional entry in the Eclipse context menu. This entry is available when the user right-clicks on a file with the .uml extension (the main file for a Papyrus model), providing a quick path to code generation.

Both plug-ins are gathered within a single Eclipse feature (for ease of installation) and are available directly from the GitHub repository. At this stage these plug-ins can be installed on the Kepler and Luna releases of Eclipse. A user manual was developed as a Wiki and is also available at the GitHub repository (https://github.com/MDDLingo/DSL3S/wiki).

4. Evaluation experiment

4.1. Experiment context

During the Spring and Summer of 2015 a user evaluation programme was conducted on DSL3S and its associated tools. An exercise was developed to expose potential users to a first contact with the language, starting with tool installation procedures and going through the development of a simple model. At the end of the exercise participants were asked to fill in an evaluation questionnaire. Various sessions were conducted where GIS professionals, computer scientists, simulation experts and environmental scientists took part.
The first of these sessions took place in May of 2015 at the Luxembourg Institute of Science and Technology with a restricted number of GIS experts as a validation of the exercise and its supporting documents. A second session took place at the same institution, this time for a broader number of participants of varied backgrounds. A third session took place at the Luxembourg Institute of Socio-Economic Research, targeting experts from other domains. Beyond these presential sessions, the test guide and questionnaire were also distributed among various electronic networks related to spatial simulation; throughout the Summer of 2015 further questionnaires were collected from these networks.

4.2. Experiment exercise
Participants followed an exercise guide that is available on-line from the DSL3S repository (http://mddlingo.github.io/DSL3S/docs/DSL3S-2015-SimpleTest-User-Guide.pdf); a static copy of the evaluation questionnaire is also available there (http://mddlingo.github.io/DSL3S/docs/DSL3S-2015-SimpleTest-Questionnaire.pdf). The guide starts with instructions to install the required Eclipse plug-ins: Papyrus, Acceleo, the DSL3S UML profile and MDD3S. Following is the installation of the required JAVA libraries, including MASON and GeoMASON, and then the download of sample spatial datasets.

The exercise evolves along 34 different steps, building a simple predator-prey model step-by-step. The user is instructed to create a Simulation and input a raster data-set for pasture with a Spatial variable. The guide then moves on to a prey Animat, introduced in the model from a sample vector data-set and constituted by an Attribute for internal energy (that declines with time). Various Operation elements are then used to model the prey seeking the best pasture with a Move element, grazing with an Harvest element and replicating once a certain energy threshold is reached using a Replicate element. Life enduring conditions are set with a Perish element. Predator comes next, another Animat element, whose instances are set randomly in space at simulation start; it is also constituted by a declining energy Attribute. The Predator Animat is modelled with a Move element to seek prey, eat prey through an Harvest operation, replicate in conditions similar to Prey, and constrained to exist only if its internal energy is positive (through a Perish element). At this point the Prey Animat is modified to avoid predators with a second Move element. As a final step, the user is instructed to input an additional spatial data-set as a Spatial variable defining areas where prey cannot enter using a third Move element. Table 1 summarises the exercise.

Figure 3 shows a run of the complete model described in the exercise guide. Blue dots are prey, red dots are predators and green areas represent pasture availability, from dense (dark green) to sparse (pale green); grey geometries represent inaccessible areas. A detailed description of this model and an overview of its dynamics can be found in de Sousa and Silva (2015).

Each session started with a 15 min long presentation on DSL3S introducing its main concepts. Participants were then given 50 min to complete the exercise.

4.3. Participants
The individuals invited to the test sessions either had experience in GIS, spatial analysis or simulation. They all possessed some level of understanding on spatial data (e.g. vector and raster data types). This is a population of individuals that, if not yet, may come to have contact with spatial simulation, therefore being potential users of DSL3S and the MDD3S infrastructure.

In total 19 individuals participated in the evaluation programme, all submitting valid responses to the questionnaire. The majority where men, mostly with ages between 25 and 35 (Figure 4 top). This is a typical distribution in a Computer Science dominated population. More than half of the participants hold a MSc title, with a third of the sample composed by PhDs. This is also reflected in a high percentage of researchers composing the sample (Figure 4 middle).
About half of the participants declared to be primarily trained in Computer Science, although only one third declared being professionals in the field. The number of those declared as researchers or faculty members is also almost half, indicating a sample not so reliant on pure software engineers.

### 5. Results
Beyond the answers to the questionnaires, performance aspects were recorded throughout each exercise; they provide a different kind of information on ease of use and are presented in Section 5.1. The results collected during these sessions concern in first place the impressions of participants on the language and the accompanying tools. Section 5.2 presents these results through a series of descriptive statistics, while Section 5.3 explores differences between sub-populations of the participants through inference statistics.

#### Table 1. Summary of the tasks performed during the evaluation exercise

| Phase               | No. | Task                                                                 |
|---------------------|-----|----------------------------------------------------------------------|
| Tool set up         | 1   | Install plug-ins                                                     |
|                     | 2   | Set up Java libraries                                                |
|                     | 3   | Obtain spatial datasets                                              |
| Project set up      | 4   | Create project                                                       |
|                     | 5   | Set spatial datasets                                                 |
|                     | 6   | Create model                                                         |
|                     | 7   | Apply DSL3S profile                                                  |
|                     | 8   | Create simulation element                                            |
| Basic prey          | 9   | Create pasture element                                               |
|                     | 10  | Create prey                                                          |
|                     | 11  | Add prey view                                                        |
|                     | 12  | Define Prey state                                                    |
|                     | 13  | Set prey grazing pasture                                             |
|                     | 14  | Transform model to code and run                                      |
| Refine prey         | 15  | Set Prey replication                                                 |
|                     | 16  | Set prey perishing                                                   |
|                     | 17  | Set prey movement                                                    |
|                     | 18  | Transform model to code and run                                      |
| Predator            | 19  | Create predator element                                              |
|                     | 20  | Add predator view                                                    |
|                     | 21  | Define predator state                                                |
|                     | 22  | Set predator replication                                             |
|                     | 23  | Set predator perishing                                               |
|                     | 24  | Set predator movement                                                |
| Animat interactions | 25  | Add interactions view                                                |
|                     | 26  | Create Supply element                                                |
|                     | 27  | Create Harvest element                                               |
|                     | 28  | Associate Harvest with Supply                                        |
|                     | 29  | Transform model to code and run                                      |
| Inaccessible areas  | 30  | Create Inaccessible Areas element                                    |
|                     | 31  | Add inaccessible areas to prey view                                  |
|                     | 32  | Modify prey movement                                                 |
|                     | 33  | Transform model to code and run                                      |
|                     | 34  | Modify layer order (if needed)                                       |
5.1. Performance
In every testing session blocking issues were logged within each different phases of the exercise (as per Table 1). As additional information, participants were also enquired on how far they reached along the 34 tasks composing the exercise.

The blocking issues registered during the various sessions concentrate in the early phases of the exercise, as Figure 5 shows. A common problem was the lack of an up-to-date Java execution environment installed on the system—various participants had Java version 7 or earlier installed and started the exercise without upgrading. Fortunately, the upgrade process is rather swift. In some cases, the installation of the DLS3S plug-ins was hampered by a slow internet connection. A smaller number of blocking issues is also attributable to a lack of familiarity with Eclipse and its graphical metaphors, e.g. in the modelling phase, applying the profile to an empty model was not obvious to all participants.

Only 3 of the 19 participants were able to complete all the tasks composing the exercise within the given time, as shown in Figure 6. Two tasks stand out as most frequent points of exercise completion, numbers 15 and 28. Task 15 corresponds to the first model-to-code transformation; in task 28 is modelled a supply-harvest relationship between the two animats. However, apart from one exception, all participants were able to: (i) install the DSL3S plug-in and required libraries; (ii) create at least an embryo of the final model, and (iii) apply the transformation at least once and run the resulting Java implementation.

Many participants spent half or more of the exercise time span in the tool install and project set-up phases (tasks 1 to 8). This again points to these being the most hurdling phases. In general, once the modelling phase was reached progress promptly accelerated. It is important to note that tool installation is a one-time process, while project set-up is mostly shaped by the Eclipse framework, therefore, these phases are to a great extent external to the language itself.
5.2. Descriptive statistics

The questionnaire required the rating of various aspects in a Likert type scale of one to five. This scale ranged from least to highest suitability, with two negative answers, one neutral and two positive. The meanings attributed to each rate were: 1—Very Low; 2—Low; 3—Neutral; 4—High; 5—Very
Figure 5. Distribution of blocking issues.

Figure 6. Exercise completion distribution.

Table 2. Summary statistics for the language questions

|                      | $\nu$ | $\mu$ | $\sigma$ | $\gamma_1$ |
|----------------------|-------|-------|----------|------------|
| Number of concepts   | 3.00  | 3.32  | 1.00     | 0.01       |
| Notation             | 3.00  | 3.58  | 0.69     | 0.68       |
| Ease of learning     | 4.00  | 3.79  | 0.54     | $-2.25$    |
| Suitability          | 3.00  | 3.37  | 0.68     | $-0.54$    |
High. This type of scale has shown to be more sensitive with smaller samples (Sauro & Dumas, 2009), as is the case. The aspects evaluated were spread across three main areas: (i) the DSL3S language, (ii) the MDD3S tool support and (iii) the usefulness of this MDD approach.

Regarding the language, the questionnaire posed the following questions:

(i) How suitable is the number of concepts in the language?
(ii) How easy to use is the UML Profile notation?
(iii) How easy is to learn the language?
(iv) How suitable is the language for the spatial simulation development domain?

Table 2 and Figure 7 summarise the results for these questions, presenting the median ($\bar{y}_2$), mean ($\mu$), standard deviation ($\sigma$) and skewness ($\gamma_1$) of the distribution. The ratings are somewhat spread regarding the number of concepts captured by the language, but in general tend more towards the positive. Language notation is deemed positive more clearly and ease of learning receives an almost unanimous high rating. As for suitability to the spatial simulation domain, results are less clear; however, they are still mostly positive.

Table 3. Summary statistics for the tool usability questions

| Tool Type          | $\bar{y}$ | $\mu$ | $\sigma$ | $\gamma_1$ |
|--------------------|-----------|-------|----------|-------------|
| DSL3S plug-ins     | 4.00      | 4.00  | 0.97     | -1.46       |
| Papyrus            | 4.00      | 3.69  | 1.02     | -0.84       |
| Code generation    | 5.00      | 4.44  | 0.86     | -1.43       |
| Whole              | 4.00      | 3.94  | 0.80     | -0.56       |
Tool usability results are summarised in Table 3 and Figure 8. The questions posed were:

(i) How do you rate the usability of Eclipse with the DSL3S and MDD3S plug-ins?
(ii) How do you rate the usability of the Papyrus Model Editor?
(iii) How do you rate the usability of the code generation method?
(iv) How do you rate the development process as a whole?

Of all the matters inquired, these were the aspects receiving the highest ratings from the participants. The DSL3S plug-ins and the model-to-code transformation mechanism received clearly high ratings, with grade five within the fourth quartile. Papyrus is also evaluated positively, but about a rate level lower. On the set of tools as a whole the results cluster around a high positive rating, with three outliers spread around.

Concerning the MDD approach, the questions posed were:

(i) How much can DSL3S help domain experts lacking programming skills prototyping their own simulations?
(ii) How much can DSL3S or a similar approach help in communicating spatial simulation models to stakeholders?
(iii) Can DSL3S or an MDD approach be the basis for a standard language to describe spatial simulation models, considering in particular communication with peers?
(iv) Would you consider such a tool for development or prototyping of your own spatial simulation projects?

Table 4 and Figure 9 portray the results for these questions, that are less conclusive. There is a slightly positive trend, but definitely not as marked as for the other aspects evaluated. Only in communication with peers a higher positive rating can be devised. In all questions the first standard deviation below the mean extends down to negative rates.

Figure 8. Mosaic diagram for the tool usability questions.
5.3. Inference statistics
These results were further explored with the goal of identifying bias in the ratings furnished by participants of different backgrounds. In a first analysis, participants were subdivided in two sub-populations: the participants trained in Computer Science and those not. In a second analysis, the participants were divided into the participants experienced on spatial simulation and those inexperienced. In these analyses the $\chi^2$ test was applied, a classical method employed to assess statistical hypothesis (Heumann, Schomaker et al., 2016). In this case it was used to appraise deviations between the answers of the sub-populations described above. The null hypothesis is that rates are similar between the two sub-populations, the $\chi^2$ test evaluates whether such is the case. In the tables presented in this Section are also reported the degrees of freedom (df) and the standard probability value (p-value) that provides a further assessment point on the extent to which the hypothesis explains the observation.

The sample is sub-divided almost in equal numbers between computers scientists (ten) and participants with other backgrounds (nine); results of the test for these two sub-populations are presented in Table 5. In 11 of 12 questions the $\chi^2$ values obtained are relatively high, resulting in probability values well above the conventional 0.05 significance level (Cowles & Davis, 1982). Therefore, for these questions the null hypothesis cannot be rejected, i.e. the answers provided by computer scientists are similar to other participants.

|                      | $\nu$ | $\mu$ | $\sigma$ | $\gamma_1$ |
|----------------------|-------|-------|----------|------------|
| Lacking skills       | 3.00  | 3.47  | 0.91     | 0.29       |
| Communication        | 4.00  | 3.63  | 0.76     | -0.78      |
| Standard language    | 3.00  | 3.21  | 0.86     | 0.13       |
| Own development      | 3.00  | 3.21  | 0.79     | -0.35      |

The sample is sub-divided almost in equal numbers between computing scientists (ten) and participants with other backgrounds (nine); results of the test for these two sub-populations are presented in Table 5. In 11 of 12 questions the $\chi^2$ values obtained are relatively high, resulting in probability values well above the conventional 0.05 significance level (Cowles & Davis, 1982). Therefore, for these questions the null hypothesis cannot be rejected, i.e. the answers provided by computer scientists are similar to other participants.

Table 4. Summary statistics for the general approach questions

![Mosaic diagrams for the general approach questions.]
The exception is the question on the possibility of applying this methodology to produce a standard language in the spatial simulation domain. With a $\chi^2$ value above 12 and a probability level of $5.9 \times 10^{-3}$, the null hypothesis is clearly rejected. Figure 10 presents the answers provided by both sub-populations. Computer scientists appear more inclined to consider the methodology as the basis for a standard language, with a majority of positive answers; it is, however, a more diverse sample. On the other hand, non computer scientists answered mostly neutrally, showing a larger degree of scepticism towards this question.

In the second analysis, the number of participants is less balanced with seven experienced on spatial simulation against twelve without such experience. The results of the $\chi^2$ tests for these two sub-populations are gathered in Table 6; again the null hypothesis cannot be rejected in most questions. The question on the suitability of the methodology for a standard language is once more an exception, with a clear rejection of the null hypothesis. Another question, the evaluation of the development process as a whole, obtains a value so close to the convention threshold that deserves further investigation too.

It is interesting to note a similar divide for the question on the basis for a standard language among these two sub-populations (Figure 10 right). Participants experienced in spatial simulation provide more consistent answers, in majority neutral. Non-experienced participants are more diverse in their answers to this question, but with a trend towards positive ratings.

Regarding the development process as whole, a rift between the two populations is not as clear (Figure 11). Participants experienced on spatial simulation deliver again the most consistent results, with all answers in rating four. Non-experienced participants are more diverse, with one negative answer and three at the maximum rate. For this latter sub-population the overall appreciation is also positive; the rejection of the null hypothesis in this case roots primarily in the diversity of answers provided by non-experienced participants.

The uniformity of answers across the four sub-populations tested is noticeable, indicating that, overall, user appreciation is not biased by background or previous experience.

### 6. Interpretation

These results are not straightforward to interpret and to some extent can even be considered contradictory. The language itself is clearly rated in a positive way. This is likely an outcome of the relatively small number of stereotypes; if it causes some doubts regarding adequacy, it also renders the
language easy to learn. Interestingly, even though specific icons have not yet been included with the UML profile plug-in, notation is also deemed positively adequate. This later aspect possibly hints at the information contained in the stereotype properties themselves being more relevant to users than graphical expression per se. The less clear orientation regarding language suitability is a first hint of some degree of scepticism towards the MDD methodology.

The tools developed are rated unequivocally high. Model-to-code transformation is a single click event whose output of multiple Java classes was even surprising to some users. As for Papyrus, the UML editor, it has improved greatly in usability the past few years, with many bugs fixed and new functionality introduced. The exercise execution performance hints at some burdens with installation, even if the Eclipse plug-in system can be regarded as an easy path to deploy an MDD infrastructure. Notwithstanding, participants did not judge these difficulties as crucial.

### Table 6. χ² test results for participants experienced with spatial simulation vis-à-vis those not experienced

|                        | χ²  | df | p-value |
|------------------------|-----|----|---------|
| **Language**           |     |    |         |
| Number of concepts     | 4.414 | 4  | 0.3528  |
| Notation               | 1.318 | 2  | 0.5173  |
| Ease of learning       | 0.735 | 2  | 0.6924  |
| Suitability            | 0.855 | 2  | 0.6523  |
| **Tools**              |     |    |         |
| DSL3S plug-ins         | 4.535 | 3  | 0.2092  |
| Papyrus                | 0.662 | 3  | 0.8821  |
| Code generation        | 3.158 | 3  | 0.3679  |
| Whale                  | 9.164 | 3  | 0.0272  |
| **Approach**           |     |    |         |
| Lacking skills         | 1.428 | 3  | 0.6991  |
| Communication          | 3.242 | 3  | 0.3558  |
| Standard language      | 7.719 | 3  | 0.0522  |
| Own development        | 1.042 | 2  | 0.5939  |

Figure 10. Results for computer scientists vs. others (left) and experienced vs. non experienced in spatial simulation (right) regarding the Standard language question.
The evaluation of the approach itself is far less conclusive. In general, there is a degree of mistrust towards the idea of developing models with graphical constructs instead of coding. There is perhaps an element of scepticism regarding application scope in these results.

Inference statistics tests were conducted to assess potential bias between different sub-populations of the participants. For the sub-populations of computer scientists and other training backgrounds, results for all questions showed a high degree of dependency, meaning that both sub-population issued similar appreciations. The question on the hypothesis of MDD providing a standard language for the domain was the only exception. Computer scientists delivered a far more positive answer to this question, whereas participants with other backgrounds left mostly a neutral appreciation. In all likelihood, the lack of familiarity with MDD (and possibly UML) is at the root of this bias between the two sub-populations.

The same tests were conducted for two other sub-populations, the participants experienced on spatial simulation and those not experienced. Once again a high degree of dependency was found for most questions, except two. The question related to the basis for a standard language pointed again to independent answers. Experienced participants showed mostly neutral, while non-experienced participants were more positive. This scepticism from experienced participants might be the reflex of some concern on becoming constrained with a limited language; it can also be the fear of losing control over the source code (something hinted at in informal exchanges with participants).

In what is one of the most bewildering and encouraging results of this study is the clearly positive appreciation of the development process as a whole by participants experienced with spatial simulation. However, it is hard to square this result with the scepticism towards MDD as basis for a standard. This dissonance points to a scepticism more of a subjective nature than born out of effective practical difficulties.

In sum, these results show that something is definitely to gain with the approach proposed with DSL3S. However, MDD appears alien to some participants, raising a degree of resistance to adoption.
MDD is yet to become mainstream and is still missing in many curricula, including Computer Science. On the other hand, the role of MDD as a means to improve communication between computer scientists and field experts seems still widely unrecognised.

7. Conclusions
In order to evaluate DSL3S and its supporting tools, various controlled test sessions were conducted in which potential users—mostly spatial analysts and computer scientists—were exposed for a first time to the tools developed. These users were overall satisfied with the language at the conceptual level and praised the ease of development adduced by DSL3S. However, some scepticism prevails regarding the MDD approach in general, still perceived as an alien methodology, weakening the prospects of a wider adoption of solutions such as DSL3S.

Beyond an exercise in GIS or computing, DSL3S is an experiment on the interfacing between computer science and a technical field of application. Contrary to most other engineering fields, Computer Science can rarely create value by itself. Its real power emerges when it is applied to practical domains, be it through automation, systematic monitoring, calculus, simulation, etc. If computing is today correctly perceived as an indispensable tool in any practical science, its teaching in these areas has remained largely limited to the inclusion of introductory programming courses. The potential of MDD to improve communication between software engineers and stakeholders or domain experts appears not yet fully acknowledged by the Computer Science community itself. For the time being, it is mostly taught and employed as an alternative programming paradigm.

Not only regarding the results of this work, but recalling the success of languages such as SysML and ModellicaML, it is possible to envision an untapped potential for MDD as an interface paradigm in general. The remarkably uniform results obtained in this exercise, with a largely similar appreciation among computer scientists and participants of diverse backgrounds, point firmly in this direction. The need to promote MDD among traditional engineering and practical science domains becomes therefore evident. Far more important than learning programming basics, is for modern day engineers to learn the appropriate tools to formally engage and co-develop with computer scientists. In this sense, MDD certainly justifies further investment from the Computer Science community, namely with the consolidation of open source MDD enabling tools.

Acknowledgements
The authors would like to thank all the participants in the evaluation sessions and Dr Benoît Otjacques (Luxembourg Institute of Science and Technology) and Dr Hichem Omrani (Luxembourg Institute of Socio-Economic Research) for facilitating their conduction.

Funding
The authors received no direct funding for this research.

Author details
Luís Moreira de Sousa
E-mail: luis.de.sousa@protonmail.ch
ORCID ID: http://orcid.org/0000-0002-5851-2071
Alberto Rodrigues da Silva
E-mail: alberto.silva@acm.org

1 ISRIC - World Soil Information, Droevendaalsesteeg 3, Building 101, 6708 PB Wageningen, The Netherlands.
2 INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Rua Alves Redol, 9 1000-029 Lisboa, Portugal.

Citation information
Cite this article as: Usability evaluation of the domain specific language for spatial simulation scenarios, Luís Moreira de Sousa & Alberto Rodrigues da Silva, Cogent Engineering (2018), 5: 1436889.

References
Berryman, M. (2008). Review of software platforms for agent based models. Science And Technology.
Challenger, M., Kardas, G., & Tekinerdogan, B. (2015). A systematic approach to evaluating domain-specific modeling language environments for multi-agent systems. Software Quality Journal, 1–41.
Cowles, M., & Davis, C. (1982). On the Origins of the.05 Level of Statistical Significance. American Psychologist, 37(5), 553–558.
de Sousa, L. M., & Silva, A. R. (2011). Review of spatial simulation tools for geographic information systems. In Proceedings of the Third International Conference on Advances in System Simulation (SIMUL 2011). ThinkMind.
de Sousa, L.M. & Silva, A.R. (2015). Showcasing a domain specific language for spatial simulation scenarios. In Proceedings of the LV European Regional Science Association Congress - ERSA.
de Sousa, L. M. & Silva, A. R. (2016). A Domain Specific Language for Spatial Simulation Scenarios. GeoInformatica, (2011), 117–149.
Degenne, P., Lo Seen, D., Parigot, D., Forax, R., Tran, A., Ait Lahcen, A., Curé, O., & Jeansoulin, R. (2009). Design of a Domain Specific Language for modelling processes in landscapes. Ecological Modelling, 220, 3517–3535.
Ermentrout, G. B., & Edelstein-Keshet, L. (1993). Cellular Automata Approaches to Biological Modeling. Journal of Theoretical Biology, 160, 97–133.
Ewais, A., & De Troyer, O. (2014). A Usability Evaluation of Graphical Modelling Languages for Authoring Adaptive 3D Virtual Learning Environments. In CSEDU, 1, 459–466.

Franklin, S., & Grasser, A. (1997). Is an agent or just a program? A taxonomy for autonomous agents. In Intelligent Agents III: Agent Theories, Architectures, and Languages (pp. 21–35). Berlin: Springer-Verlag.

Grignard, A., Toillardier, P., Gaudou, B., Vo, D., Huynh, N. Q., & Drogoût, A. (2013). GAMA 1.6: Advancing the art of complex agent-based modeling and simulation. G. Boella, E. Elkind, B. T. R. Savarimuthu, & F. Dignum (Eds.), In PRIMA 2013: Principles and Practice of Multi-Agent Systems (Vol. 8291 of Lecture Notes in Computer Science, pp. 117–131). Berlin Heidelberg: Springer. doi:10.1007/978-3-642-44927-7_9

Heumann, C., Schomaker, M., & et al., (2016). Introduction to statistics and data analysis: With exercises, solutions and applications. In R. Chap (Ed.), Association of two variables (pp. 67–96). Springer.

Luke, S., Cioffi-Revilla, C., Panaït, L., Sullivan, K., & Balan, G. (2005). “MASON: A Multi-Agent Simulation Environment.” Simulation: Transactions of the society for Modeling and Simulation. International, 82(7), 517–527.

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

Morais, F., & Silva, A. R. (2015). Assessing the quality of user-interface modeling languages. In Proceedings of ICESS/2015 Conference. SCITEPRESS.

Railsback, S. F., Steven, L. L., & Jackson, J. K. (2006). Agent-based simulation platforms: Review and development recommendations. Simulation, 82(9), 609–623.

Saura, J., & Dumas, J. S. (2009). Comparison of three one-question, post-task usability questionnaires. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI ’09 (pp. 1599–1608). New York, NY: ACM. doi:10.1145/1518701.1518846

Selic, B. (2003). The pragmatics of model-driven development. IEEE Software, 20(5), 19–25.

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

Silva, A. R. (2015). Model-driven engineering: A survey supported by a unified conceptual model. Computer Languages, Systems & Structures. Retrieved from http://www.sciencedirect.com/science/article/pii/S1477842415000408

Wilson, S. W. (1991). The animat path to AI. In J. A. Meyer & S. Wilson (Eds.), From animals to animats (pp. 15–21). Cambridge, MA: MIT Press.