A Description Language for Environmental Fields Synthesis in Maritime Co-Simulation Scenarios

Liqun Wu, Axel Hahn
Department of Computing Science
Carl von Ossietzky Universität Oldenburg
Oldenburg, Germany
e-mail: {liqun.wu, axel.hahn}@uni-oldenburg.de

***Abstract*—This paper introduces a Domain Specific Language (DSL) for describing the physical environment required in maritime co-simulation scenarios. This language helps simulation scenario modelers intuitively describe the physical environment which they wish their vessels to act in. The modelers can write programs to specify which environmental elements should be present in a simulation scenario, and how these elements should vary within the spatio-temporal span of the scenario. An environment synthesis framework is built based on this language. It uses programs written in this language to compose dynamic environmental fields. Thus, scenario modelers can focus on the simulation models they want to explore, while still have full control of the environment in their simulation scenarios.**

Keywords-Domain Specific Language; maritime co-simulation; environmental data synthesis; spatio-temporal field.

I. INTRODUCTION

Safety has been a significant challenge in constructing maritime traffic systems [1]. This is also true when applying eNavigation technology and assistance systems for navigators. Risk assessment of these systems is important to improve the safety of maritime eNavigation [2]. Research reveals that co-simulation platforms are appropriate for the assessment via running simulation scenarios. The co-simulation platforms distribute simulation tasks to different components. Each component focuses on one specific aspect of the simulation. All the components work collaboratively to accomplish the whole simulation. In this context, a vessel simulation component simulates vessel objects moving on the water. These objects need to receive information of influential environmental elements during the run of simulation scenarios, e.g., current and wind. The required environmental elements are assumed to be provided by another environment synthesis component, therefore vessel modelers can focus on modeling the behavior of vessels. The environment discussed here is particularly referred to the simulated physical environment in which the objects are situated and whose evolution is beyond the control of the objects [3].

To supply the information of the environmental elements, the environment synthesis component could either make use of existing observation datasets, or run a calculation model based on initial settings. However, edge cases, e.g., extreme weather conditions cannot be fully covered, although testing various conditions is crucial to assess safety issues. Furthermore, vessel modelers cannot fully control and modify the behavior of the environmental elements in simulation scenarios. Thus, they need a machine processable solution to specify the environment matching their scenarios. This solution should not require too much effort from vessel modelers that will cause distraction from their focuses on modeling vessels.

This paper proposes the Environmental Fields Description Language (EFDL) to describe the user-controlled environmental elements. It is a DSL that allows user to specify environmental element fields based on a simple and intuitive field structure. It aims to provide the formal language support for describing environment from requirement perspective. The scenario modelers can express what kind of physical environment their scenarios are situated in with domain abstractions in this language. Together with the EFDL, an environment synthesis framework is also proposed for composing dynamic environmental data that matches the EFDL description during simulation runs.

The rest of this paper is organized as follows: Section 2 illustrates the related work on DSL in simulation domain and remained problems; Section 3 introduces the conceptual structure and benefits of the proposed language, followed by Section 4 that describes how to implement and use this language in an environment synthesis framework. Section 5 concludes with discussion and future work.

II. RELATED WORK

DSLs are computer languages specialized for certain domains [4]. A DSL has higher expressiveness than general computer languages in its target domain that allows domain experts to program with domain abstractions. Various guidelines have been developed for design and implementation of DSLs [4][5][6]. Modern Language Workbenches [7][8][9] assist in the development of DSLs.

DSLs have gained attention in spatio-temporal modeling and simulation. Maxwell and Costanza [10] have proposed the spatial modeling language (SML) to support modular simulation design. It mainly focuses on structuring and linking different modules for a model. SELES [11] landscape modeling framework provides a language to specify the model dynamics by defining how an event affects its neighborhood. It supports various raster data as input. Ocelet [12] uses an

 ISBN: 978-1-61208-501-2

unusual approach that takes Service-Oriented Architecture (SOA) concepts to model processes in landscape. It relies on the evolving of the coded cases of the model actions, but does not include vocabularies to specify the spatio-temporal aspects.

More recent development takes the advantages of the model-driven methods and language workbenches. GAMA platform [13] provides a modeling language GAML to support the development of agent-based models. The GAML semantic is defined in a meta-model and implemented using XText [14] in a textual manner. It is an agent-oriented, typed language which allows to agentify Geographic Information System (GIS) data [15]. Environment can be specified by the modeler by a list of geometries in addition to being loaded from files and created as agents. It also supports to write equations linked to agent attributes imperatively. DSL3S [16] supports prototyping spatial simulation models using graphic notation. Its implementation utilizes the Eclipse Modeling Framework (EMF)-based graphic editor and an open source code generator. The DSL3S abstract syntax is defined as UML profiles, with each construct defined as a stereotype. This is a general level specification language with only 11 concepts. The "Spatial" concept provides means to load spatial environmental data files for the simulation.

All the above-mentioned DSLs focus on the description of the phenomenon that the simulation aims to explore. While most of them assume that the influential environmental information is available, GAML allows to input a list of user-defined environmental geometries. Nevertheless, it treats the influential environmental elements as agents, makes no conceptual distinction from the phenomenon that the simulation explores. It also does not emphasize the field-form environment, which is the common case in maritime domain. The researchers who developed ELMS [17] have realized the needs of the environment specification. They proposed the ELMS as a language to specify the simulated environment in multi-agent simulations. The objects which the agents' percept are defined as resources. However, their focuses remain in social simulation. Although the grid representation of the environment is mentioned, no means of defining resource variation among cells is provided.

III. ENVIRONMENTAL FIELD DESCRIPTION LANGUAGE

This paper proposes the EFDL to express the required change of the environmental elements in the simulated environment which the model objects act in. For each maritime simulation step, vessels perceive influential environmental elements, e.g., wind. Then, the received information is used as input to derive next actions of the model objects. Such an environmental element forms a field covering the space which the model objects move through and should be provided to the model objects during the simulation timespan, even at some locations its values could be zero or null. Vessel modelers also need to modify the variation of environmental fields matching different scenarios to test how their models react under different conditions. The EFDL allows modelers to write a formal description of these required environment fields with domain abstractions.

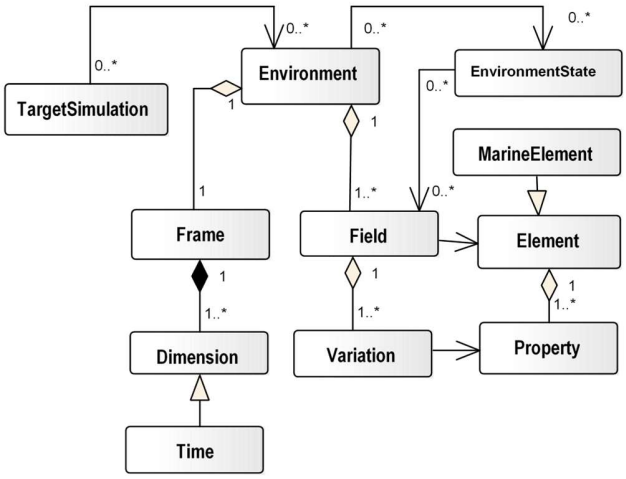

Figure 1. Conceptual structure of an environmental description.

The conceptual specification of the language has three modules: *Environment*, *Variation* and *Marine*. The *Environment* module defines the core structure of a description of the environment. The conceptualization is presented in Figure 1 leaving out the supporting concepts and attributes. It is explained in following paragraphs with examples of the formal definitions of the concepts, as well as a simple descriptive example of an environment with wind presented, required by scenarios to test how a 2D positioning system for maritime crafts endures different wind strengths.

The entry concept *Environment* is defined as a tuple consisting of one *Frame* and probably one or more *Fields*, formally as:

$$\begin{aligned} &\text{Environment} := \langle \text{frame, fields} \rangle, \\ &\quad \text{frame} \in \text{Frame, fields} \subset \text{Field.} \end{aligned} \tag{1}$$

All fields have a common *Frame* that represents the spatio-temporal span within which the target scenario is supposed be performed. Even if an element does not appear at some location, the user should be able to define this unavailability. *Frame* comprises of *Dimensions*. It is formally defined as in (2). The presented supporting function, i.e., *orthogonal: Dimension*×*Dimension*→*Boolean*, takes two instances of *Dimension* as input and returns true if they are orthogonal to each other.

$$\begin{aligned} &\text{Frame} := \langle \text{dimensions} \rangle \\ &\text{where dimensions} \subset \text{Dimension,} \\ &|\text{dimensions}| \in \{1,2,3,4\}, \\ &\forall\, d_i, d_j \in \text{dimensions: orthogonal}\left(d_i, d_j\right) = \text{true.} \end{aligned} \tag{2}$$

Dimensions in EFDL are limited to the physical spatio-temporal space. Each *Dimension* represents the spread of the *Frame* in one direction in the space. *Time* is defined as a subtype of the *Dimension* which is orthogonal to any spatial dimension. This concept is formally defined as following:

$$\text{Dimension} := \langle \vec{d}, \text{origin} \rangle. \tag{3}$$

The vector $\vec{d}$ represents the direction of the *Dimension*. After all, the *Frame* is conceptually an orthogonal spatio-temporal frame. The *Frame* is anchored by the *origins* of its *Dimensions*. The full definitions of the supporting concepts are omitted here. In practice, user can declaratively align one *Dimension* to an axis of common coordinate systems and define the *origins* using coordinates in the aligned system.

In our example, three *Dimensions* can be defined in the *Frame*, two of which are aligned to the axes of the WGS84 system. The other one is the time dimension referenced to the UTC+0 time. The defined *Frame* is grounded by setting the *origins* as the WGS84 coordinates of the most south-west point of the simulation region and the start time of the simulation. Having the above-introduced concepts, this can be done by simple declarations of instances of corresponding concepts and value assignments to the instances.

A *Field* represents the influential field of one environmental element whose information needs to be received by the model objects in the target scenario. Each *Field* is associated with an environmental *Element* and consists of one or more *Variations*. We define the supporting function *binding: Element×Field* to associate the *Field* to the corresponding *Element*. Each *Variation* in this field is associated with a *Property* of the *Element*. It defines how the values of the corresponding property should distribute. The formal definitions of the binding function and its input types described in this paragraph are represented in (4) leaving out the definitions of the support concepts. A similar function to associate the Variation to the Property is also provided.

$$
\begin{aligned}
&\text{Field} := \langle \text{variations} \rangle \\
&\text{where} \quad \text{properties} \sqsubset \text{Variation}. \\
&\\
&\text{Element} := \langle \text{properties, name} \rangle \\
&\text{where} \quad \text{properties} \sqsubset \text{Property, name} \in \text{String}. \\
&\\
&\text{binding: Element} \times \text{Field} \\
&\text{binding(element, field)} := \\
&\forall \text{ variation} \in \text{field:} \\
&\text{propertyOf(variation)} \in \text{element} \wedge \\
&\text{elementOf(field)} = \text{element}.
\end{aligned}
\tag{4}
$$

Our example includes one *Field* (denoted as *f*) associated with *wind* element via *binding (f, wind).* The *wind* element is defined in the *Marine* module which will be described in a later paragraph. The *Field f* has two *Variations* representing wind properties, namely the speed of x direction and of y direction. They are also defined in the *Marine* module.

Variation module express how the numerical values of an element property should fill the corresponding *Frame*. It does not focus on an accurate derivation based on available information, but on the easy expression of desired conditions. *Variation* can be defined in groups of one dimension (e.g., time) or of more dimensions (e.g., two spatial dimensions). Correlated groups can be further combined. Such a group is defined as a *Subvariation* consisting of a reference to the corresponding dimension(s) and a function with the coordinates in the dimension(s) as variables:

$$\text{Subvariation} := \langle \text{ref.dimensions, F(ref.dimensions)} \rangle \tag{5}$$

Currently, we test two *Variation* types: pattern association is used to declare a typical pattern of an environmental element, which is pre-defined in the *Marine* module; customized grid allows user to divide the *Frame* into blocks by declaring the number of cells and the cell size along each dimension. The gridded blocks (cells) imply the desired resolution of the fields. Then the values distribution can be expressed with integer values based on the indexes of cells. As a simple illustration, the modelers could fix both wind speed values of cells with index 1 in time dimension (e.g., via mapping to a raster file) and declare that all the values should be linearly increased by a factor of 2 in time dimension.

An additional concept *EnviromentState* is used to specify which element values should be provided at each simulation step. It is defined in relation to the state of vessels such as their spatio-temporal location, neighborhood and offset. Most simply, user can state that all the element values at current vessel location should be provided to the vessels.

The *Marine* module is an application specific module with common types of involved environmental elements, such as the *wind* in the example. All the concrete environmental elements in this module are instances of the *MarineElement*, which itself is a subtype of the *Element*. It also includes typical patterns formation of these elements, which the *Variation* can refer to. Its separation from the other two core modules enhances the extensibility of this language. Further elements or application-specific modules can be added without modifying the core conceptualization of the simulated environment.

The EFDL brings three main benefits. First, EDFL descriptions are more intuitive from a "required condition" perspective. It distinguishes the controlled environmental variation from the phenomenon to be explored. This distinction avoids the need to transfer the conceptualization of environment into the conceptualization of active models similar to what the users investigate on, as some previously mentioned DSLs did. Second, it avoids the miscommunication between the component which requires information of certain environmental elements and the component which provides this information. As a computer language, the description written in the EFDL will be interpreted only in one way as defined in the semantic specification of the language. Third, the EFDL enables the user to control the environmental elements. It goes beyond indicating a dataset input directory and supports the users to describe their desired variation. This is especially useful when running scenarios with diverse conditions, which may require synthesis environmental fields.

IV. IMPLEMENTATION APPORACH AND ENVIRONMENT SYNTHESIS FRAMEWORK

The implementation of the EFDL uses an "abstract syntax first" approach. It means that the concepts and data structure holding the information of a program [6] as described in

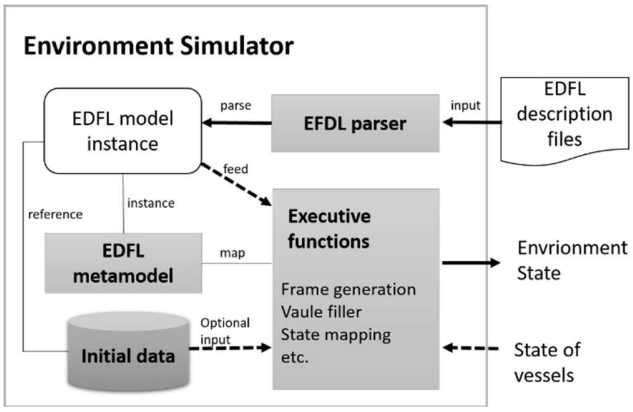

Figure 2. Environment synthesis framework.

Section 3 will be implemented at first. The concrete syntax containing the notations which is actually used to write the DSL program [6] will be defined based on the abstract syntax. While this language is domain-specific and usually needs to be combined with other general level tools to build a complete application, the most suitable notation format may be various in different situations. This way of implementation can have multiple concrete syntax that may needed for different applications without modifying the conceptualization in the abstract syntax.

We use the Eclipse Modeling Framework (EMF) [18] to define the abstract syntax. All the concepts and conceptual structure described in Section 3 will be stored in Ecore metamodels which can be easily integrated with widely-used EMF-based tools. A prototype editor will be developed as an Eclipse plug-in based on the metamodels using EMFText [19]. EMFText is an EMF-based Language Workbench that can define the textual syntax of DSLs. The grammar rules and the function signatures built on top of the conceptualization are defined in the concrete syntax using EMFText. EMFText maps them to the abstract syntax, i.e., the Ecore metamodels. The concrete syntax is used directly by users. Using the textual concrete syntax, users describe fields that are analogous to their description sentence in mind. The textual programs are also easier to be quantitatively compared with programs written in general purpose language for evaluation. The editor will be used to write the EFDL program in the concrete syntax.

The EFDL forms the base of our environment synthesis framework as shown in Figure 2. The EFDL editor comes with an execution engine that will be embedded in the environment synthesis component which we call environment simulator. It is one part of our maritime co-simulation platform. To run a scenario, the environment simulator loads a corresponding EFDL program. It parses the loaded program as an EFDL model which is an instance of the language metamodel. Then the content of this model is consumed by the corresponding executive functions to produce values of environmental elements. The values are mapped to the spatio-temporal locations specified in the *Frame* of the program model. Any data referenced in the model instance are also used by these functions. At each simulation step, the engine produces relevant element values based on current time and states (e.g., locations) of the vessel objects, as defined in the *EnvironmentState* of the model.

V. CONCLUSION AND FUTURE WORK

Although various DSLs have been developed for spatio-temporal simulations, they focus on the phenomena to be investigated and assume the environmental elements that influence these phenomena to be given. This is not always true, especially for the safety critical simulations that have to run under various conditions. The modelers require the environment matching their scenarios. The EFDL proposed in this paper enables modelers to fully control the influential environmental elements by describing how they wish their fields vary using this language with an intuitive expression structure and domain abstractions. An environment synthesis framework is designed based on the EDFL to process the description written in this language and to compose the described environment for simulation scenarios.

This language separates the field structure and its value variation from the application-specific environmental elements. While the current usage is to support maritime co-simulations, the core field specification can be reused for simulations in other application domain combined with an application vocabulary extension. The implementation of this language takes advantage of language workbenches and allows multiple concrete syntax that adapted to different applications.

Currently, we are investigating on various formations of the element variations. The aim is to find the balance between the language simplicity and its expressiveness. This language should have the ability to express the environment with complicated change. The basic idea is to decompose its variation into simple facets (i.e., *the Subvariations*). Thus, how to correctly describe the combination of these facets in an easy way become crucial. It needs also to be investigated that to what extend the environment can be described under the conceptualization of the "field with a frame" as introduced in Section 3. When combining with the *Marine* module to express a specific environment, if all the environmental elements can be mapped into this conceptualization, or if additional conceptualization is needed, remain as open questions.

The developed language and framework will be tested with vessel modelers in the future by a series of test cases including generation of spatial varied (e.g., current), temporal varied fields (e.g., wind changing strength over time) and spatio-temporal varied fileds (e.g., current changing over time). The participating modelers will be asked to fill a questionnaire regarding the usability based on the test cases. The program length of these test cases will be compared to the program length of the same cases written in general purpose programing language to evaluate its productivity, following the approach from Kosar et al. [20].

The performance of the implemented simulator is beyond the scope of the research on the formal specification of the required environment description. However, it is important from the engineering point of view. While the specification is independent from the algorithms that creating values for

 ISBN: 978-1-61208-501-2

specific fields, the efficiency of the simulator depends also on the chosen algorithms. The computational cost of the above-mentioned test cases should be also evaluated.

ACKNOWLEDGMENT

We thank the ministry of science and culture of Lower Saxony for supporting us with the graduate school Safe Automation of Maritime Systems (SAMS).

REFERENCES

[1] H. Gale and D. Patraiko, "Improving navigational safety - The role of e-Navigation," *Seaways*, pp. 4–8, Mar. 2007.

[2] C. Läsche, V. Gollücke, and A. Hahn, "Using an HLA Simulation Environment for Safety Concept Verification of Offshore Operations," in *Proceediings 27th European Conference on Modelling and Simulation*, Ålesund, Norway, pp. 156–162, 2013.

[3] F. Klügl, M. Fehler, and R. Herrler, "About the Role of the Environment in Multi-agent Simulations," in *Proceedings of First International Workshop, E4MAS 2004*, New Yourk, NY, USA, pp. 127–149, 2004.

[4] M. Mernik, J. Heering, and A. Sloane, "When and how to develop domain-specific languages," *ACM Comput. Surv. CSUR*, vol. 37, no. 4, pp. 316–344, Dec. 2005.

[5] G. Karsai, H. Krahn, C. Pinkernell, B. Rumpe, M. Schindler, and S. Völkel, "Design guidelines for domain specific languages," in *Proceedings of 9th OOPSLA Workshop on Domain-Specific Modeling (DSM' 09)*, Helsinki School of Economics. TR no B-108. Orlando, Florida, USA, arXiv:1409.2378, 2009.

[6] M. Völter, *DSL Engineering: Designing, Implementing and Using Domain-Specific Languages*. CreateSpace Independent Publishing Platform, 2013.

[7] M. Fowler, "Language Workbenches: The Killer-App for Domain Specific Languages?," 2005. [Online]. Available: http://www.martinfowler.com/articles/languageWorkbench.html. [Accessed: 01-Mar-2016].

[8] B. Merkle, "Textual modeling tools: overview and comparison of language workbenches," in *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, Reno/Tahoe, Nevada, USA, pp. 139–148, 2010.

[9] S. Erdweg, T. van der Storm, M. Völter, and M. Boersma, "The state of the art in language workbenches: conclusions from the Language Workbench Challenge," in *Proceedings of the International Conference on Software Language Engineering (SLE, 2013)*, USA, pp. 197–217, 2013.

[10] T. Maxwell and R. Costanza, "A language for modular spatio-temporal simulation," *Ecol. Model.*, vol. 103, no. 2–3, pp. 105–113, Nov. 1997.

[11] A. Fall and J. Fall, "A domain-specific language for models of landscape dynamics," *Ecol. Model.*, vol. 141, no. 1–3, pp. 1–18, Jul. 2001.

[12] P. Degenne, D. L. Seen, D. Parigot, and R. Forax, "Design of a domain specific language for modelling processes in landscapes," *Ecol. Model.*, vol. 220, no. 24, pp. 3527–3535, 2009.

[13] P. Taillandier, D. A. Vo, E. Amouroux, and A. Drogoul, "GAMA: A simulation platform that integrates geographical information data, agent-based modeling and multi-scale control.," in *Proceedings of the 13th International Conference on Principles and Practice of Multi-Agent Systems*, Kolkata, India, pp. 242–258, 2010.

[14] M. Eysholdt and H. Behrens, "Xtext: implement your language faster than the quick and dirty way," in *Proceedings of the ACM international conference companion on Object oriented programming systems languages and applications companion*, Reno, Nevada, USA, pp. 307–309, 2010.

[15] A. Grignard, P. Taillandier, B. Gaudou, D. A. Vo, N. Q. Huynh, and A. Drogoul, "GAMA 1.6: advancing the art of complex agent-based modeling and simulation," in *Proceedings of PRIMA 2013: Principles and Practice of Multi-Agent Systems*, vol. 829, pp. 117–131, 2013.

[16] L. de Sousa and A. R. da Silva, "A domain specific language for spatial simulation scenarios," *GeoInformatica*, vol. 20, no. 1, pp. 117–149, Jan. 2016.

[17] F. Okuyama, R. H. Bordini, and A. C. da Rocha Costa, "ELMS: An Environment Description Language for Multi-agent Simulation," in *Proceedings of First International Workshopt on Agent Environments for Multi-Agent Systems*, New Yourk, NY, USA, pp. 91–108, 2004.

[18] D. Steinberg, F. Budinsky, M. Paternostro, and E. Merks, *EMF: Eclipse Modeling Framework*, 2nd ed. Addison-Wesley Professional, 2008.

[19] "EMFText." [Online]. Available: http://www.emftext.org/index.php/EMFText. [Accessed: 01-Mar-2016].

[20] T. Kosar, P. E. M. López, P. A. Barrientos, and M. Mernik, "A preliminary study on various implementation approaches of domain-specific language," *Inf. Softw. Technol.*, vol. 50, no. 5, pp. 390–405, Apr. 2008.

 ISBN: 978-1-61208-501-2