A roadmap to domain specific programming languages for environmental modeling

Key requirements and concepts

Ioannis N. Athanasiadis
Electrical & Computer Engineering Dept
Democritus University of Thrace
Xanthi, Greece
ioannis@athanasiadis.info

Ferdinando Villa
Basque Centre for Climate Change (BC3)
IKERBASQUE, Basque Foundation for Science
Bilbao, Spain
ferdinando.villa@bc3research.org

ABSTRACT

The limited reuse of current environmental software can be blamed in part on the tools used to develop it; the use of generic-purpose programming languages makes it particularly hard. As environmental scientists strive to prioritize the clear statement and communication of the semantics of natural systems in favor of understanding software implementations of their models, Domain-Specific Languages may come to help, offering the option of truly declarative environment for environmental modeling. This paper discusses some key requirements and concepts for developing Domain-Specific Languages that can inform and streamline environmental modeling, and previews some use scenarios using examples from a DSL in development.

Keywords

Environmental modeling, Domain specific languages, Applied computing Environmental sciences, Ecoinformatics, Software interoperability and reuse, Semantic modeling

1. INTRODUCTION

Environmental assessment and ecosystem service valuation are calling for integrated tools that can be relatively easily improved, and making them more transparent to a wide audience, varying from decision makers to the general public [1]. In our knowledge-driven society, science remains hidden in environmental data and software, while still requires to become openly available. At the same time environmental decision-making is required to become more transparent, supported by both evidence (data) and arguments (models).

We argue that significant limitations of the current systems and practices arise from the tools used, and specifically from generic-purpose programming languages. Environmental scientists have strived for many decades to express environmental data and theory with general-purpose tools, and practice has proved that this is not a straightforward task. Nature's complexity seems not to be easily expressed with mainstream programming models that have become prevalent during past decades. Experience drives us to conclude that existing languages and tools make statements too complex for environmental problems, as they introduce pleonasms, redundancies, and boundaries in verbose expressions about Nature, putting a spoke in science's wheel. Seems like environmental modeling is trapped by tools that don't match the needs, a situation that resembles the *golden hammer* anti-pattern [2].

To improve this situation, new paradigms and tools are required for determining how complex problems are perceived, formalized, and communicated. The compartmental study of Nature may lead us to biased forecasts and false conclusions, thus sound model integration is required more than ever to address complex issues (i.e. how climate change may affect food security).

In this paper, we study the key requirements for developing Domain-Specific Languages (DSL) for environmental modeling and discuss pros and cons. A DSL that could directly encode modeling knowledge into software artifacts would have a tremendous impact on model integration and collaborative science, by enabling the automation and verification of the compositions, opening new pathways for the future.

In the following Section 2 we present methods and tools for our research: We present environmental modeling and software concepts, and current practices researches employ for reusing environmental software. Then we present how the environmental domain imposes key requirements for software reuse, and what was achieved with knowledge-driven approaches. We also provide with a brief overview of DSL practices. Then in Section 3 we present key requirements for a DSL for environmental modeling and present a couple of examples in the form of user stories. The paper concludes with a discussion on expected benefits of our approach.

2. METHODS AND TOOLS

2.1 Environmental models and frameworks

Environmental *models* are scientific abstractions of Nature and its behavior, and *environmental software* allows them to be used for computerized simulation, optimization or decision support. Software implementations are the tools of the scientists for answering such questions through *in-silico* experiments for *ex-ante* assessments. Efficient management and reuse of environmental software is the key to maximize the returns of environmental modeling investments and ensure that domain complexity is treated effectively.

When an environmental model is encoded in a programming language, new limitations are introduced compared to the original modeling assumptions. Hardly ever these assumptions can be represented directly in the implementation language of choice; on the contrary, this knowledge resides with the modelers. Employing procedural abstraction as binding contract is rarely employed effectively, and documenting these specifications happens rarely in practice. To make things worse, during the last decades, a number of models have been designed and implemented, and it has become natural to assemble them together in order to address more complex problems than the original. Environmental modeling and software are challenged to deal with complexity, uncertainty, scaling and integration issues, all qualities inherited from the physical world.

Integrated assessments are becoming increasingly common in environmental management and therefore scientists are faced with the problem of integrating models across scales and disciplines. This is not a straightforward process, and integration of environmental software is not the sole necessary condition for a proper assemblage of models, and credible science. Today, integrated modeling is mostly focused on the mechanics of the integration, through computerized e-science tools for managing data and software to assist scientists with the technical linking of models to create scientific workflows [3, 4, 5, 6, 7].

2.2 Current practices: frameworks for reuse

Environmental modeling frameworks (e.g. CCA [8], TIME [9], ESMF [10], OpenMI [11], and OMS [12]) written in general-purpose programming languages, as Fortran, C, MATLAB, Java, or GAMS, are used by scientists to implement environmental models, and typically offer an Application Programming Interface (API) that implements routines tailored to environmental modeling needs.

While they allow the management, reuse and integration of mathematical models from various disciplines and at different spatio-temporal scales, these solutions tend to be heavyweight and maintenance-intensive [13]. Also, the actual reusability of models is often left to the modeler's own discipline and responsibility to structure a complex model with smaller reusable components [14].

On top of this, modeling knowledge is very poorly incorporated into software implementations. Model interfaces embrace a critical amount of the modelers knowledge, but their software implementations are *poor reflections* of modelers' perceptions, as they fail to represent the complexity of model assumptions in software terms [15]. Many environmental models relate to the structure of the systems they represent only partially. In fact, very few models can serve as an explanation of the modeled processes: the understanding of the system is usually implicit and typically resides outside the model specification and implementation. As a result, the purpose of models is typically restricted to the specific application they have been developed for; the potential for reuse, communication and integration with data and other models is limited [16]. In the environmental modeling community, often it is perceived to be easier to (re)create a new model than to take an existing one and adapt it to new needs [17].

An environmental model, as an abstraction aimed to an expressive and parsimonious representation of the complexity of the real world, approaches its subject from a specific point of view, reflecting simplifying assumptions about the phenomena involved. Therefore, it simplifies the full extent of causal chains and driving forces of the phenomena of interest, to the benefit of simplification, focalization and modularization. Environmental software implementations introduce even more limitations (for instance, data ranges may be discretized).

Environmental software states the assumptions made for building it only in an implicit manner, which requires experts to be understood. For instance, the spatial discretization of a model variable can only be inferred by a close inspection of the data type used to implement it [15]. Furthermore, for integrated assessments, existing environmental software needs to be integrated across scales and disciplines. For example, the AgMIP initiative aims to integrate agricultural models with climate scenarios.

2.3 Domain introduces requirements for reuse

Sound integration and reuse of existing models and software is a bottleneck for integrated modeling activities, and key factor of the overall project success. This resonates with priorities in Software Engineering, that promotes the concepts of reusing components off-the-shelf [18], distributed computing [19], agent-based computing [20], service-oriented architectures and web services [21] to support the development of modular applications. The very same concepts are meant to be used to develop modular and integrated environmental software applications.

However, software integration is not the sole necessary condition for proper assemblage of environmental models. If a set of (good) environmental software components is working together, this is not at all a sign that the compound model will make any sense from a modeling point of view and generates credible results. Prior research has targeted quality assurance issues in the integration of environmental models [22, 23], but focused mainly on the quality of the modeling process, not the role of the software. General-purpose software engineering methods for component integration are not enough to ensure sound integration for enabling collaborative science.

Environmental software may (and should) embody sophisticated statements of environmental knowledge. Yet, the knowledge it incorporates is rarely self-contained enough to be understood and used –by humans or machines– without the modeler's mediation [16]. Inspired by the declarative programming approaches that became popular in the 80's, *declarative modeling* has been suggested as a remedy for the "black box" nature of self-contained models [24, 25, 26, 27]. Graphical languages, as Simile [28] and STELLA [29], support for fairly self-explanatory model statements and greatly enhanced readability of model components. However, they have mostly remained focused on syntactic aspects rather than semantics, thus making them unsuitable for large-scale model reuse. Model integration in software terms does not guarantee sound integration of the model logics [15] and declarative modeling is no exception on this trend [16].

2.4 Incorporating domain knowledge

A possible remedy to the problem appears to be *knowledge-based computing*, which requires a paradigm shift in the way environmental scientists think about modeling.

Semantic modeling in environmental sciences demonstrated its potential with the *mediation approach*, where formal knowledge is the key to automatic integration of datasets, models and analytical pipelines. The next step, applied on experimentally at this stage, is the *knowledge-driven approach*, where the knowledge is the key not only to integration, but also for overcoming scale and paradigm differences, and automated knowledge discovery. For an in depth discussion refer to [30].

Despite the clear potential offered by *semantic modeling* for environmental sciences, only limited case studies are available, i.e IMA [31], ESD [32], SEEK [5, 33], and ARIES [30]. The feasibility of wide adoption of the approach remains to be seen in the coming years. However, without suitable tools and languages, the paradigm shift required would carry a cost too significant for widespread adoption.

Domain-specific programming languages (DSLs) may become important in achieving these goals. The success of a DSL for environmental modeling depends on its capacity to formalize and incorporate the semantics of natural systems, to unify representations of data and metadata, improve their usability in scientific workflows, and ease the definition and composition of dynamic models.

2.5 Domain-specific languages: concepts and tools

Domain-specific languages are programming languages tailored to a specific application domain. '*They offer substantial gains in expressiveness and ease of use compared with general-purpose programming languages in their domain of application, with corresponding gains in productivity and reduced maintenance costs*' [34]. DSLs are considered '*enablers of reuse*', as they offer several benefits over APIs, including appropriate domain-specific syntax and notation, constructs and abstractions, and built-in functionality for analysis, verification, optimization, parallelization, and transformation (AVOPT).

DSLs do not just hide complex designs of general-purpose programming languages, and impose good coding practice to their adopters, but implement structures and offer features that are not straightforward to implement with general-purpose programming languages. DSLs overcome the shortcomings of API libraries, such as limited domain-specific notations and the inability of domain-specific analysis, verification, optimization and transformation that restricts their usefulness [35].

Ideally, a DSL follows the domain abstractions and semantics as closely as possible, letting developers perceive themselves as working directly with domain concepts [36]. The past few years, DSLs have become a popular trend in software engineering, also thanks to new methodologies and tools that allow developing them with ease [36]. For example, Xtext [37] covers all aspects of a complete language infrastructure, from parser through linker, compiler or interpreter, to fully functional integrated development environments.

3. DSL FOR ENVIRONMENTAL SOFTWARE

3.1 Key requirements

A Domain-Specific Language for environmental modeling represent a step ahead of the state-of-the-art in environmental software. By incorporating domain knowledge into programming constructs, it may offer a programming environment which will essentially turn environmental modeling into a scientific activity '*as it once was*'. Environmental scientists equipped with a DSL are enabled to concentrate on their domain-specific modeling problems, letting implementation issues to be taken care of the programming language environment.

Several Domain-Specific Language for environmental modeling are already (or soon will be) developed. Some may focus on certain disciplines, others may target specific modeling paradigms or frameworks. As there is an underling layer of common needs presented above in Section 2, these languages may incorporate some of following features:

(R.1) *Domain-specific data structures* that describe units and quantities, accuracy, spatial and temporal scales and extents, quality and provenance information of data sources and results. Separate logical models of observable entities and their observations can be used to enable a novel, sophisticated approach to semantic representation of environmental data sets.
These developments may be founded on previous work for defining semantics of environmental terminology as SWEET [38, 39] and ARIES [30], and adopt Dublic Core standards [40] for Open Archives principles [41] for metadata sharing.

(R.2) *Semantic annotation of interfaces* decorated with rich metadata, that incorporate model assumptions, pre- and post- conditions, and prerequisites for reuse, in machine-readable formats, with the goal to support model chaining in scientific workflows.
This may take advantage of previous work on decorating model interfaces, either as components or services, as discussed in [17, 42, 43, 44].

(R.3) *Typical operations*, such as scaling, averaging, interpolation or unit conversions, should be intrinsic features of the language, not requiring user attention through method calls.
For example, the language may be able to intelligently aggregate quantities across differently scaled observations, properly distinguishing between intensive and extensive quantities.

(R.4) *Support for different modeling paradigms*, able to be cross-compiled for different *environmental modeling frameworks*, ensuring backwards compatibility. Common modeling paradigms as system dynamics, probabilistic modeling, agent-based modeling, and black-box modeling (as bayesian, or neural networks), need to be considered.

(R.5) *Account for modeling uncertainty* and quality information, through build-in computations with confidence intervals (or distributions) in order to incorporate different sources of uncertainty (i.e. random sampling error and biases, noisy or missing data, approximation techniques for equation integration, projections of alternative futures, etc).
For example, standard error propagation may be built in the language: given two variables x and y , represented as mean/variance pairs $(\mu_x, \sigma_x^2)$and (μ_y, σ_y^2) respectively, their difference $x-y$ should be calculated $(\mu_x - \mu_y, \sigma_x^2 + \sigma_y^2)$

(R.6) *Model transparency and defensibility of results* support with visualization to justify model results. Results may be associated with a history of operations on original sources, that documents its provenance, and document property rights, or quality assurance.
This may build upon previous work on provenance models for e-science using Semantic Web tools [45] or the Open Provenance Model [46].

Last but not least, an appropriate Integrated Development Environment (IDE) needs to support the use of the language, comprised from a coding and a graphical environment. Also, a very important component of success for the success of such an endeavor is the development of training material and documentation, that need to account from simple to complex models and for diverse categories of end-users ranging from scientists with no-programming experience to experienced environmental modelers.

3.2 Use scenarios

In the past three years we have developed a DSL primarily targeting ecosystem service valuation [47], as the enabling infrastructure for the ARIES framework [30]. The language, still in development, exhibits many of the characteristics we discussed in the previous section, and we use it below for exemplifying some scenarios of use, in the form of user stories.

Story 1: Semantics for data annotation
Consider an environmental scientist that has put together a dataset on buffalo population. Assume that data has been stored in some format; it may be a website, a database, FTP or some other service. Its annotation with rich semantics may look like the following example:

```
model wcs(service = "http://eco.logismi.co/wcs",
               id = "BF")
  named buffalo
  as count livestock:Buffalo per km^2
  with metadata {
    dc:description "FAO Gridded Livestock Dataset"
     dc:rights "cc-attr-nomod"
     dc:source "http://www.fao.org/..."
     im:distribution "public" };
```

While the statement above resembles the annotation of a web service, it is far more expressive, as it contains three major parts: *a*) the *query part* that specifies how to technically retrieve the data from a WCS server, *b*) the observation semantics part which links these data to a concept of a livestock ontology and declares units, and spatial and temporal reference, *c*) the metadata part that contains information on resource provenance, licensing and reuse.

The language framework is equipped with domain-specific data structures, that enable the application of typical operations as unit conversion, scaling, interpolation, etc, and the semantic mediation among different schemata. Common operations as retrieving a WCS layer from a map server is abstracted and connection implementation is left to the language. At the same time, it accounts for quality information and provenance metadata, while it has clear contracts for reusing this dataset.

Story 2: Simple modeling activities
Simple environmental models are often defined via simple methods as:
a) rule modification from an existing model, which can essentially be considered an argument passing or a filter operation,
b) specification from a generic model, which resembles subclasses with the factory pattern.

Consider that the flood risk in some study was defined by a simple rule, as:

```
model SimpleFloodRiskModel
  as classification im.risk:FloodRisk
 observing (im.geo:Elevation in m) named elevation
  on definition
          set to 'low' if [elevation > 100]
                 'med' if [elevation <= 100
                           and elevation > 20]
                 'high' if [elevation<=20]
```

In the examples above, model inputs and outputs are associated with concepts in an ontology (respectively `FloodRisk` and `Elevation`), which enables the language system to subsequently compose and reuse the model with automated type checking, validate model chains and composition. This may be enabled by reasoning on the model semantics using Description Logics [48].

In a second example, consider a case where models may be applied dynamically, at runtime, based on a rule that refers to data values, or spatiotemporal context as for example in:

```
define SOUTHERN_ROCKIES
  as space(shape = "EPSG:4326
                  POLYGON((-109.25 41.25, ...");

model im.soil:SurfaceErosion
  observing (Slope as measure im.geo:Slope in °)
     named slope
  as
    models.soil-loss-equation-model
                              if slope < 9.17,
    models.bayesian.soil-erosion-steep-usa
                           if in SOUTHERN_ROCKIES,
    models.bayesian.soil-erosion-steep-global
                              otherwise;
```

The model rule is declared as a case statement that incorporates semantically rich *'variables'* that refer to concepts in an ontology and have units, or to regions defined as polygons, thus allowing the language system to apply the model on any dataset that provides with the same concept, and mediate with typical operations as scaling or unit conversion. At the same time, the model is prescribed in a declarative fashion, that allows the language system to reform it into a query, and search for data that satisfy the model requirements.

While the language being developed is far more expressive in its full capacity, extending to the handling of multiple scales of space/time in an agent-based paradigms, it is beyond the scope of this paper to provide with more details at this stage. Readers shouldn't hesitate to contact the authors for further details.

4. EXPECTED BENEFITS & DISCUSSION

The case studies discussed exemplify the interdisciplinary and multidisciplinary character that may determine the success of environmental DSLs. Joining perspectives in software engineering and environmental modeling in a language, and harnessing the combined powers of knowledge representation, artificial intelligence and ontologies, such efforts may help open new perspectives and integration opportunities in environmental sciences and ecology. Besides the benefit of introducing a new integrative paradigm for environmental modeling to the environmental sciences, agriculture and ecology, it is also aimed to:

i) benefit the software engineering community by evaluating existing methodologies and test patterns for DSL development;
ii) provide a performant, yet realistic testbed for DSL engineering bringing forth issues of performance, parallelization and distributed computing;
iii) advance the notion of *semantic modeling* as a new paradigm for environmental modeling where all concepts used to model natural systems are explicitly defined by ontologies.

Even if DSL development is considered a hard undertaking, requiring both domain and language development expertise, the state of the art in DSL development methodologies is been advanced enough to support domain specific design [49]. From our experience so far, we realized that DSL development is not a simple sequential process: '*preliminary analysis may have to supply answers to unforeseen questions arising during design, and design is often influenced by implementation considerations*' [34]. However, these problems is well attained by the software engineering researchers and several patterns to address them have been identified, that cover the whole process from domain analysis, to DSL design, implementation and deployment, as the methodology of *Mernik et al (2005)* [34], and open source tools for DSL development as [37] can be proved useful.

We acknowledge that it is extremely important to engage early domain scientists in the language development, and we have done so by organizing an annual Spring University on Ecosystem Service modeling in Bilbao[1], during which we introduced the language, had training sessions and got feedback on its development from our students, who used it for modeling their course projects. This proved to be an excellent testbed both for the design and the implementation of our DSL.

The time has come for environmental scientists to **need to invest less in understanding software implementations** of their models, and focus more in the semantics of natural systems in order to produce useful models. During the past 40 years, environmental modeling efforts, using general-purpose languages led to a babel of software components that are used only by their owners and have no capacity for reuse. Knowledge resides with the modelers, and software is useless without the heavy intervention of experts, that *interpret* it. A DSL may offer the first truly declarative environment for environmental modeling, that will go beyond the model syntactics and account for semantics.

Acknowledgments

Dr Athanasiadis was partially supported by the ALPINE project funded by the Greek Secretariat of Research and Technology (grant 11SYN-6-91), and the AgMIP network.
The ARIES project which partially funded this research for Villa was originally funded by the US National Science Foundation (grant 9982938) and receives support from the ASSETS project funded by ESPA/NERC (grant NE-J002267-1).

[1]For more on the International Spring University on Ecosystem Service modeling see the school website: `http://www.bc3research.org/springuniversity/`

5. REFERENCES

[1] J. Rotmans, "Methods for Integrated Assessment: The challenges and opportunities ahead," *Environmental Modeling and Assessment*, vol. 3, pp. 155–179, 1998.

[2] W. H. Brown, R. C. Malveau, and T. J. Mowbray, *AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis*. Wiley, 1 ed., Mar. 1998.

[3] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pp. 423–424, IEEE, 2004.

[4] E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, *et al.*, "Pegasus: A framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming*, vol. 13, no. 3, pp. 219–237, 2005.

[5] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger-Frank, M. Jones, E. Lee, J. Taoand, and Y. Zhao, "Scientific Workflow Management and the Kepler System," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 10, pp. 1039–1065, 2006.

[6] S. Lee, T. D. Wang, N. Hashmi, and M. P. Cummings, "Bio-STEER: A Semantic Web workflow tool for Grid computing in the life sciences," *Future Generation Computer Systems*, vol. 23, pp. 497–509, 2007.

[7] S. Cuddy and P. Fitch, "Hydrologists Workbench - a hydrological domain workflow toolkit," in *Proc. 5th Intl Congress on Environmental Modelling and Software (iEMSs 2010)*, iEMSs, 2010.

[8] R. Armstrong, D. Gannon, A. Geist, K. Keahey, S. Kohn, L. McInnes, S. Parker, and B. Smolinski, "Toward a common component architecture for high-performance scientific computing," in *Proc 8th Int'l Symp on High Performance Distributed Computing*, pp. 115–124, 1999.

[9] J. Rahman, S. Seaton, J.-M. Perraud, H. Hotham, D. I. Verrelli, and J. Coleman, "It's TIME for a new environmental modelling framework," in *MODSIM 2003 International Congress on Modelling and Simulation*, pp. 1727–1732, 2003.

[10] N. Collins, G. Theurich, C. DeLuca, M. Suarez, A. Trayanov, V. Balaji, P. Li, W. Yang, C. Hill, and A. da Silva, "Design and Implementation of Components in the Earth System Modeling Framework (ESMF)," *International Journal of High Performance Computing Applications*, vol. 19, no. 3, pp. 341–350, 2005.

[11] M. Blind and J. Gregersen, "Towards an Open Modeling Interface (OpenMI): The HamonIT project," *Advances in Geosciences*, vol. 4, no. 1, pp. 69–74, 2005.

[12] L. Ahuja, J. Ascough II, and O. David, "Developing natural resource modeling using the object modeling system: feasibility and challenges," *Advances in Geosciences*, vol. 4, no. 1, pp. 29–36, 2005.

[13] W. Lloyd, O. David, J. A. II, K. Rojas, J. Carlson, G. Leavesley, P. Krause, T. Green, and L. Ahuja, "Environmental modeling framework invasiveness: Analysis and implications," *Environmental Modelling & Software*, vol. 26, no. 10, pp. 1240 – 1250, 2011.

[14] A. E. Rizzoli, M. Donatelli, I. N. Athanasiadis, F. Villa, and D. Huber, "Semantic links in integrated modelling frameworks," *Mathematics and Computers in Simulation*, vol. 78, no. 2-3, pp. 412–423, 2008.

[15] I. N. Athanasiadis, A. E. Rizzoli, M. Donatelli, and L. Carlini, "Enriching environmental software model interfaces through ontology-based tools," *International Journal of Applied Systemic Studies*, vol. 4, no. 1-2, pp. 94–105, 2011. Accepted for publication.

[16] F. Villa, I. N. Athanasiadis, and A. E. Rizzoli, "Modelling with knowledge: a review of emerging semantic approaches to environmental modelling," *Environmental Modelling and Software*, vol. 24, no. 5, pp. 577–587, 2009.

[17] D. P. Holzworth, N. I. Huth, and P. G. de Voil, "Simplifying environmental model reuse," *Environmental Modelling & Software*, vol. 25, no. 2, pp. 269 – 275, 2010.

[18] A. Egyed, H. Muller, and P. D. E., "Integrating COTS into the development process," *IEEE Software*, vol. 22, no. 4, pp. 16–18, 2005.

[19] H. Attiya and J. Welch, *Distributed Computing: Fundamentals, Simulations, and Advanced Topics*. Willey, 2nd edition ed., 2004.

[20] M. Luck, P. McBurney, and C. Preist, eds., *Agent Technology: Enabling Next Generation Computing, A Roadmap for Agent Based Computing*. AgentLink, 2003.

[21] T. Erl, *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice-Hall, 2004.

[22] J. C. Refsgaard, J. P. van der Sluijs, J. Brown, and P. van der Keura, "A framework for dealing with uncertainty due to model structure error," *Advances in Water Resources*, vol. 29, no. 11, p. 1586–1597, 2006.

[23] A. Jakeman, R. Letcher, and J. Norton, "Ten Iterative Steps In Development and Evaluation of Environmental Models," *Environmental Modelling & Software*, vol. 21, no. 5, pp. 602–614, 2006.

[24] D. Robertson, A. Bundy, R. Muetzelfeldt, M. Haggith, and M. Uschold, *Eco-Logic. Logic-based Approaches to Ecological Models*. MIT Press, 1991.

[25] V. Wenzel, "Semantics and syntax elements of a unique calculus for modelling of complex ecological systems," *Ecological Modelling*, vol. 63, no. 1-4, pp. 113–131, 1992.

[26] R. M. Keller and J. L. Dungan, "Meta-modeling: a knowledge-based approach to facilitating process model construction and reuse," *Ecological Modelling*, vol. 119, no. 2-3, pp. 89–116, 1999.

[27] R. Muetzelfeldt, "Declarative Modelling in Ecological and Environmental Research," Position Paper EUR 20918, European Commission Directorate-General for Research, European Commission, Brussels, Belgium, 2004.

[28] R. Muetzelfeldt and J. Massheder, "The Simile visual modelling environment," *European Journal of Agronomy*, vol. 18, pp. 345–358, 2003.

[29] B. Richmond, *An introduction to systems thinking: STELLA software*. High Performance Systems Inc, 2001.

[30] F. Villa, M. Ceroni, K. Bagstad, G. Johnson, and S. Krivov, "ARIES (ARtificial Intelligence for Ecosystem Services): a new tool for ecosystem services assessment, planning, and valuation.," in *11th International BIOECON Conference*, 2009.

[31] F. Villa, "Integrating Modelling Architecture: a declarative framework for multi-paradigm, multi-scale ecological modeling," *Ecological Modelling*, vol. 137, pp. 23–42, 2001.

[32] F. Villa, M. Ceroni, and S. Krivov, "Intelligent Databases Assist Transparent and Sound Economic Valuation of Ecosystem Services," *Environmental Management*, vol. 39, pp. 887–899, 2007.

[33] J. Madin, S. Bowers, M. Schildhauer, S. Krivov, D. Pennington, and F. Villa, "An ontology for describing and synthesizing ecological observation data," *Ecological Informatics*, vol. 2, no. 3, pp. 279–296, 2007. Ecology; Observation; Measurement; Ontology; Data discovery; Data integration.

[34] M. Mernik, J. Heering, and A. M. Sloane, "When and How to Develop Domain-Specific Languages," *ACM Computing Surveys*, vol. 37, no. 4, pp. 316–344, 2005.

[35] T. Kosar, P. E. Martinez Lopez, P. A. Barrientos, and M. Mernik, "A preliminary study on various implementation approaches of domain-specific language," *Information and Software Technology*, vol. 50, no. 5, pp. 390 – 405, 2008.

[36] J. Sprinkle, M. Mernik, J.-P. Tolvanen, and D. Spinellis, "Guest Editors' Introduction: What Kinds of Nails Need a Domain-Specific Hammer?," *Software, IEEE*, vol. 26, pp. 15 –18, july-aug. 2009.

[37] H. Behrens, M. Clay, *et al.*, *Xtext User Guide*. The Eclipse Foundation, 2010.

[38] SWEET, "Semantic Web for Earth and Environmental Terminology (SWEET)." Available online: `http://sweet.jpl.nasa.gov`, 2006.

[39] R. G. Raskin and M. J. Pan, "Knowledge representation in the semantic web for Earth and environmental terminology (SWEET)," *Computers and Geosciences*, vol. 31, no. 9, pp. 1119–1125, 2005.

[40] "Dublin Core Metadata Initiative." Available online: `http://www.dublincore.org`.

[41] "Open Archives Initiative." Available online: `http://openarchives.org`.

[42] C. Granell, L. Díaz, and M. Gould, "Service-oriented applications for environmental models: Reusable geospatial services," *Environmental Modelling & Software*, vol. 25, no. 2, pp. 182 – 198, 2010.

[43] O. David, J. A. II, W. Lloyd, T. Green, K. Rojas, G. Leavesley, and L. Ahuja, "A software engineering perspective on environmental modeling framework design: The Object Modeling System," *Environmental Modelling & Software*, vol. 39, pp. 201 – 213, 2013.

[44] R. Knapen, S. Janssen, O. Roosenschoon, P. Verweij, W. de Winter, M. Uiterwijk, and J.-E. Wien, "Evaluating OpenMI as a model integration platform across disciplines," *Environmental Modelling & Software*, vol. 39, pp. 274 – 282, 2013. <ce:title>Thematic Issue on the Future of Integrated Modeling Science and Technology</ce:title>.

[45] J. Zhao, C. Wroe, C. Goble, R. Stevens, D. Quan, and M. Greenwood, "Using semantic web technologies for representing e-science provenance," in *The Semantic Web–ISWC 2004*, pp. 92–106, Springer, 2004.

[46] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, *et al.*, "The open provenance model core specification (v1. 1)," *Future Generation Computer Systems*, vol. 27, no. 6, pp. 743–756, 2011.

[47] F. Villa *et al.*, "Thinklab: the enabling infrastructure for ARIES." Available online: `https://bitbucket.org/ariesteam`.

[48] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, eds., *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 1993.

[49] E. Evans, *Domain-driven design: Tackling complexity in the heart of software*. Addison-Wesley, 2003.