A roadmap to domain specific programming languages for environmental modeling

Key requirements and concepts

Ioannis N. Athanasiadis\textsuperscript{1}  
Ferdinando Villa\textsuperscript{2}

\textsuperscript{1}Electrical & Computer Engineering Dept,  
Democritus University of Thrace, Xanthi, Greece  
ioannis@athanasiadis.info

\textsuperscript{2}Basque Centre for Climate Change (BC3),  
IKERBASQUE, Basque Foundation for Science, Bilbao, Spain  
ferdinando.villa@bc3research.org

13th Workshop on Domain-Specific Modeling,  
Oct 27, 2013, SPLASH, Indianapolis, IN, USA
Environmental software

- Integrated tools for environmental assessment, ecosystem service valuation, etc
  - Need for easy improvements (science and software development are mixed)
  - Wide audience (not just for science)

- Science remains hidden within environmental data and software, while still requires to become openly available

- Environmental decision-making is required to become more transparent
  - supported by both evidence (data) and arguments (models)
Environmental models and integrated modelling

- Environmental models are scientific abstractions of Nature and its behavior
- Environmental software is an implementation of environmental models to be used for computerized simulation, optimization or decision support
  - Scientists’ tools for answering questions through in-silico experiments for ex-ante assessments
- When an environmental model is encoded in a programming language, new limitations are introduced
  - Hardly ever these can be represented directly in the programming language of choice
  - On the contrary, assumptions reside with the “modeler”
  - Employing procedural abstraction as binding contract is rarely employed effectively,
  - Documenting these specifications happens rarely in practice
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 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 (i.e., Kepler, Pegasus, etc).
Environmental "Modelling Frameworks"

- Environmental modeling frameworks offer an Application Programming Interface (API) that implements routines tailored to environmental modeling needs
  - CCA, TIME, ESMF, OpemMI, OMS, etc
  - written in general-purpose programming languages

- allow the management, reuse and integration of models
  - across disciplines
  - at different spatio-temporal scales

- tend to be heavyweight and maintenance-intensive (Lloyd 2011)

- model reusability is often left to the modeler’s own discipline (Rizzoli 2008)

- they fail to represent the complexity of model assumptions in software terms (Athanasiadis et al 2011)

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 (Holzworth 2010)
What is the way forward?
Limitations of current systems arise from the tools used

New paradigms and tools are required
- to determine how complex problems are perceived, formalized, and communicated
- to address new problems (i.e. how climate change may affect food security) via sound model integration

Domain-Specific Languages (DSL) may be a remedy
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.
Domain-specific languages: concepts and tools

- DSLs 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’ (Mernik 2005)
- Offer benefits over APIs, i.e domain specific syntax and notation, constructs and abstractions
- Ideally, a DSL follows the domain abstractions and semantics as closely as possible, letting developers perceive themselves as working directly with domain concepts (Sprinkle 2009)
- DSLs are quite popular lately, and several tools and methodologies make their creation easier.
DSL for environmental software

- incorporate domain knowledge into programming constructs
- offer a programming environment which will essentially turn environmental modeling into a scientific activity ‘as it once was’.
- Environmental scientists equipped with a DSL will be enabled to concentrate on their domain-specific modeling problems, letting implementation issues to be taken care of the programming language environment.
- Several DSLs for environmental modeling are already (or soon will be) developed.
  Some may focus on certain disciplines, others may target specific modeling paradigms or frameworks
Related work

Domain introduces requirements for reuse

- **Declarative modeling** has been suggested in the late 90’s as a remedy for “black box” modeling
  - Graphical languages (STELLA, Simile) support for fairly self-explanatory model statements and greatly enhanced readability of model components
  - Mostly remained focused on syntactic aspects rather than semantics, thus making them unsuitable for large-scale model reuse

- **Semantic modeling** (IMA, SEEK, ARIES) utilizes formal knowledge for representing the semantics of environmental data and models:
  - *mediation approach*: automatic integration of datasets, models and pipelines
  - *knowledge-driven approach*: overcoming scale and paradigm differences, and automated knowledge discovery (*experimental*)
Features of a DSL for environmental modelling I

1. Domain-specific data structures
   - Units and quantities, accuracy, spatial and temporal scales and extents, quality and provenance information of data sources and results.
   - Logical models of observable entities for the semantic representation of environmental data.
   - SWEET and ARIES, DC, OAI

2. Semantic annotation of software interfaces
   - Incorporate model assumptions, pre- and post-conditions, and prerequisites for reuse
   - In machine-readable formats
   - To support model chaining in scientific workflows
   - OMS, APSIM, OpenMI, etc

3. Typical operations
   - Scaling, averaging, interpolation or unit conversions, should be intrinsic features of the language, not requiring user attention through method calls
Features of a DSL for environmental modelling II

4 Support for different modeling paradigms
   ▶ enable cross-compilation for different environmental modeling frameworks (backwards compatibility)
   ▶ system dynamics, probabilistic modeling, agent-based modeling, and black-box modeling (as bayesian, or neural networks)

5 Account for modeling uncertainty and quality information
   ▶ build-in computations with confidence intervals (or distributions)
   ▶ 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)

6 Model transparency and defensibility of results
   ▶ Associate results with a history of operations on original sources
   ▶ Document provenance, property rights, quality assurance
   ▶ Open Provenance Model, Semantic Web Tools, etc

7 Integrated Development Environment (IDE)
   ▶ coding and a graphical environment
   ▶ training material
Thinklab: a DSL for ecosystem services
Example 1: Annotate buffalo population dataset

An environmental scientist has put together a dataset on buffalo population. Assume that dataset has been stored in WCS format.

```model
cwcs(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" |
```

a. the *query part*: specifies how to technically retrieve the data

b. the *observation semantics part*: links to a concept of a livestock ontology, and declares units, and spatial and temporal reference,

c. the *metadata part*: contains information on resource provenance, licensing and reuse.
Example 2: Simple Flood Risk model

Consider that the flood risk in some study was defined by a simple rule depending on elevation

```
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]
```

- model inputs and outputs are associated with concepts in an ontology (FloodRisk, Elevation)
- enables the language system to subsequently compose and reuse the model with automated type checking, validate model chains and composition
Example 3: Model switching for surface erosion

Models may be applied dynamically, at runtime, based on a mixed rule that refers to data values, or spatiotemporal context.

```plaintext
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;
```

- **variables** refer to concepts in an ontology.
- the model can be applied on any dataset that provides with the same concept.
  - mediate with typical operations as scaling or unit conversion.
Expected benefits & discussion

- Interdisciplinary and multidisciplinary character of environmental DSLs
- Introduce a new integrative paradigm for environmental modeling to the environmental sciences, agriculture and ecology.

a. benefits for the SE community by evaluating existing methodologies and test patterns for DSL development;
b. provide a performant, yet realistic testbed for DSL engineering bringing forth issues of performance, parallelization and distributed computing;
c. 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.
Community engagement

**Early** engagement of domain scientists in the language development

**2013** First Spring University on Ecosystem Service modeling in Bilbao

- introduce the language, provide trainings and get feedback
- students used it for modeling their course projects
- got feedback on both DSL design and implementation

**2014** Second edition in preparation

http://www.bc3research.org/springuniversity/
Summary

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.
A roadmap to domain specific programming languages for environmental modeling
Key requirements and concepts

Ioannis N. Athanasiadis¹  Ferdinando Villa²

¹ Electrical & Computer Engineering Dept,
Democritus University of Thrace, Xanthi, Greece
ioannis@athanasiadis.info

² Basque Centre for Climate Change (BC3),
IKERBASQUE, Basque Foundation for Science, Bilbao, Spain
ferdinando.villa@bc3research.org

13th Workshop on Domain-Specific Modeling,
Oct 27, 2013, SPLASH, Indianapolis, IN, USA