Jul 1st, 12:00 AM

Towards integrated model building with semantically annotated components

Oliver Schmitz

D. Karssenberg

Jean-Luc De Kok

International Environmental Modelling and Software Society (iEMSs)
2012 International Congress on Environmental Modelling and Software
Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany
R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)
http://www.iemss.org/society/index.php/iemss-2012-proceedings

Towards integrated model building with semantically annotated components

Oliver Schmitz[a,b], **Derek Karssenberg**[a] **and Jean-Luc de Kok**[b]

[a] *Department of Physical Geography, Faculty of Geosciences, Utrecht University, Heidelberglaan 2, PO Box 80115, 3508 TC Utrecht, The Netherlands (o.schmitz@uu.nl, d.karssenberg@uu.nl)*

[b] *Flemish Institute for Technological Research (VITO), Unit Environmental Modelling, Boeretang 200, 2400 Mol, Belgium (jeanluc.dekok@vito.be)*

Abstract: Integrated models are valuable tools for research and decision support as they allow a comprehensive analysis of environmental systems. Component–based software frameworks aid in their development by a more straightforward construction and coupling of generic components. Formal descriptions of model components and their relationships by means of ontologies ease the clarification of the embedded knowledge. However, existing approaches do not adequately cover the development and assessment phases of integrated models. Here, we define invariant characteristics of process–based spatio–temporal systems and introduce an ontology to semantically describe model components and their interactions. We address the role of the ontology during the different model development phases. During model initialisation, the framework uses the information to check the validity of linked model components in order to prevent the exchange of mismatching information such as different data types or spatial extents. For model execution, the framework combines the semantic information of the individual model components to an integrated description. The ontology is integrated in a declarative modelling framework allowing an environmental scientist to add semantic information for component properties such as parameter values and state variables. The semantic enrichment of the model components consolidates their application in integrated systems and is a step towards improved interoperability with other modelling frameworks.

Keywords: Component–based modelling; semantic annotation; Python; PCRaster

1 INTRODUCTION

Numerical models are important instruments in research and improve the scientific understanding of complex environmental systems [e.g., Liu et al., 2002; Voinov et al., 2004], and guide the assessment of policy and management decisions in human–natural systems [e.g., Letcher et al., 2007; de Kok et al., 2010]. To represent all elements of a system under investigation, the development of numerical models increasingly requires an interdisciplinary collaboration between model developers of various disciplines and the integration of their associated domains such as hydrology or economy. The modelling process is enhanced by applying a modular development process by decomposing environmental systems into individual model components. In general, this modularisation increases the comprehensibility and eases the maintenance of individual components. Moreover, it eases the integration and thus reuse of model components outside one's field of expertise. Coupling individual model components with complementing functionality to larger systems is known as integrated modelling [Argent, 2004; Hinkel, 2009].

The development of integrated models is a challenging process as interdisciplinary knowledge and models from environmental and social sciences need to be linked [e.g., Jakeman et al., 2006; Refsgaard et al., 2007]. Depending on the objective of the study, integration of various model components needs to account for the integration of different space and time discretisation as in the coupling of model components on municipality and state level or linking processes with daily and monthly time steps, data availability and requirements, model assessment strategies such as uncertainty analysis, and the involvement of various stakeholders. As changes in the requirements and modifications of the model on the conceptual and technical level are likely, the development process needs to be traversed iteratively.

The numerical implementation of model components is supported by a broad range of tools, including domain specific languages and environmental modelling frameworks [e.g., Argent et al., 2009]. Practises adopted from computer science such as component based software engineering [c.f., Szyperski, 2002; Argent, 2004; Booch et al., 2007; Rizzoli et al., 2008] resulted in more straightforward development of generic environmental model components, and reduced the development time and increased the reliability of model components. Although these technical opportunities support individual tasks of the development cycle and ease the model development by providing domain specific functionality, the potential benefits such as reusability and interoperability are most likely applicable for individual software solutions and to a lesser extent available for different modelling frameworks. Therefore, the model developer is still in need to combine various tools in a meaningful way to achieve all tasks in the model development process. In addition to the technical interoperability problem, the numerical implementations hardly expose the conceptual and scientific knowledge embedded in the model components. Passing model components for example to the scientific community is therefore not only limited by technical but also by semantical interoperability barriers.

A way to reduce the heterogeneity barriers of model components is to add an additional knowledge representation to modelling frameworks and environmental data. This semantic knowledge can be expressed by a formal representation of concepts and their relationships with the help of ontologies [c.f., Uschold and Gruninger, 1996; Buccella et al., 2009]. Such a representation can be given in Extensible Markup Language (XML) dialects such as the Web Ontology Language [OWL, 2004]. Ontologies provide a means to standardise on technical and semantic level, and applications of semantic enrichment can be found for example in the annotation of environmental data [e.g., Saiful Islam and Piasecki, 2006; Madin et al., 2007],in data retrieval [e.g., Baglioni et al., 2008; Lutz and Klien, 2006], in the integration of web–based or service–oriented architectures [e.g, Athanasis et al., 2005; Best et al., 2007], or in ontology–based simulations [e.g., Beck et al., 2010; Janssen et al., 2011]. Semantic knowledge is also used to describe component interfaces [e.g., Moore and Tindall, 2005; Rizzoli et al., 2008] and used to extend modelling frameworks [e.g., Villa, 2007; Villa et al., 2009]. However, the construction and incorporation of semantic knowledge into environmental models adds another complexity to the development process.

In this paper, we introduce a semantic annotation layer embedded in an existing environmental modelling framework [Schmitz et al., 2012]. The framework allows environmental scientists with basic programming knowledge to develop spatio–temporal model components and to couple these to integrated systems. As the development of model components is an ongoing process, it is required that a semantic description follows these modifications directly. We therefore define the following research questions:

- Which characteristics of model components need to be formalised to enable their coupling?

- How can a formalised description still allow for a flexible development of model components?

In the following section, we introduce a formalised description for a model component as the main building block. Afterwards, we describe how the coupling of semantically enriched interfaces can be used to validate the structural setup of a coupled model. We show the prototype of the semantic layer embedded in a modelling framework that is implemented in the high–level scripting language Python, followed by an outline of further research directions.

2 COMPONENT–BASED MODELLING

2.1 Problem statement

Writing software that represents environmental systems requires large amounts of software code to map various natural and social processes over space and time. To avoid the development of single, monolithic applications with limited maintainability and extensibility, an approach by grouping functionality into model components is required. The development of complex models therefore comprises the tasks of developing these model components themselves, and to arrange these components such that the behaviour of larger systems is represented properly. It is required that flexibility exists for both tasks.

For a model builder it is preferable to use available building blocks for the process implementation of model components. Given that the developer may modify the internal working of a model component, the information about its inputs and outputs are not fixed. It is therefore required that model components come with a formal representation that adjusts when the internal working is modified.

Such formal representations of model components and connections can be expressed by means of an ontology. Ontologies are a formal explicit specification of a shared conceptualisation [Gruber, 1993] and define a common terminology for concepts and their relationships [c.f., Scholten et al., 2007]. Buccella et al. [2009], for example, review ontologies for the integration of geographic data. Formalisation approaches are applied in other domains such as in manufacturing processes [e.g., Gruninger, 2009], linguistics [e.g., Mark et al., 2007], simulation of military systems [e.g., Hofmann et al., 2011], or the philosophical inspection of concepts and their dependencies [e.g., Galton and Mizoguchi, 2009]. Comparatively, Application Programming Interfaces of programming languages are examples of formal representations. However, their expressiveness and interoperability is restricted to the used programming language. Building semantic interfaces, thus extending interfaces with domain specific meaning, leads to interfaces that are independent of the used programming language.

In the following, we define the requirement for model components and their coupling in the context of spatio-temporal model development.

2.2 Design of model components

The model component is the generic building block to represent dynamic behaviour of a subsystem. Subsystems thereby can represent elementary processes such as the slope calculation of a digital elevation map, or more complex processes such as infiltration. A model component simulates the specific processes over a set of discrete time steps t

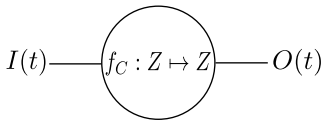

Figure 1: Structural view of a model component. The process representation f_C is encapsulated, while the inputs I and outputs O form the component interface.

according to [c.f., Beck et al., 1993; van Deursen, 1995]:

$$Z_t = \begin{cases} Z_0 & \text{for } t = 0 \\ f(Z_{t-1}, I_t, P) & \text{otherwise} \end{cases} \quad (1)$$

Here, f represents the state transition function that is constructed by the model developer and can be composed of for example differential equations, update rules or probabilistic rule sets [Karssenberg and de Jong, 2005]. I_t are the input values of the current time step t, P are the parameter values that are piecewise constant during the simulation time. Feedback is modelled by incorporating the state variables of the previous time step Z_{t-1} in the calculation of the current state variables Z_t. Note that all variables in Equation 1 are mostly spatial.

For the parameters, state and input variables a variety of data types can be used to represent process properties. For example, lumped values can be represented by data types provided by programming languages such as integer or float. Additionally, higher level constructs such as arrays or lists or user defined data types can be used. Moreover, these variables contain discretisation information such as non–spatial or spatial, or units.

Several software packages provide building blocks that can be combined in a flexible way to compose the state transition function f and thus a model component. These building blocks can originate from libraries providing support for spatial data types [e.g., GDAL, 2009] or spatio-temporal modelling and analysis [e.g., Pullar, 2003]. Here, we neglect internal implementation details and formalise the functional characteristics that are relevant to establish interoperability between model components. Approaches and application cases that incorporate formalised process descriptions can be found for example in Villa [2001], Rizzoli et al. [2008], or Beck et al. [2010].

2.3 Formalising model components

To enable interactions, model components need to obtain information about the functional characteristics of their counterparts. These functional characteristics of a model component are given by a structural and a behavioural part. The interface defines the structural characteristics of a model component and describes how interaction with other components can be established. The behavioural characteristics are given by the component initialisation and the dynamic behaviour during the simulation run (Equation 1). We therefore specify a model component by the following set of characteristics:

```
modelComponent = (interface, initialisation, dynamic)
```

Figure 1 shows the structural view on a model component and its interface. The interface provides access points to interact with other model components by exchanging data. The model component can request from and provide to several other model components. The interface is specified as a set of input and output ports:

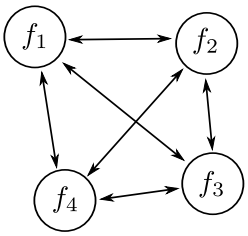

Figure 2: Assembling integrated models from building blocks. f_1, f_2, f_3, f_4 model individual processes from domains such as hydrology or economy. The model components interact via defined interfaces.

`interface = (interfacePort)`

A port of the interface is defined as

`interfacePort = (direction, variable)`

where direction indicates if a component requires an input variable or enables other components to retrieve a state variable Z as component output.

The initialisation is the basis for the dynamic behaviour of a model component and is required for an appropriate preparation of the state transition function. While the initialisation of a model component is not relevant for other components, access to initial values is required for example for calibration schemes or analysis purposes. Variables that need to be initialised are the set of parameter values, and the initial values for the state variables Z_0. We therefore specify the model component initialisation as:

`initialisation = (`Z_0`, `P`)`

Emanating from the initial state, the dynamic behaviour of a model component is modelled by the repeated execution of the state transition function by iterating over a set of time steps (Equation 1). As the specification of the state transition function is considered as encapsulated, we only capture the temporal extent and discretisation of the model component. With the temporal characteristics it is possible to access state variables of a model component regardless of the way they are calculated. The dynamic characteristics are given by the temporal extent and discretisation:

`dynamic = (start, end, `Δt`)`

with start and end time defining the simulation horizon and the time step discretisation Δt.

2.4 Model design

With the model components specified in the previous section the model builder can use these as building blocks to construct models (Figure 2). Here, coupling involves the definition of relationships between model components. As model components only specify the interface, the model developer needs to define the relationships between model components by connecting output and input ports. Defining relationships performed by the model developer in a manual step can be error prone such as in the case of coupling model components using different units for their variables. It is therefore required that the modelling framework automatically evaluates the relationships defined by the modeller

```
<modelComponent>
  <comment>simple storage calculation</comment>
  <interface>
    <interfacePort>
      <direction>input</direction>
      <variable>
        <name>precipitation</name>
        <spatialType>spatial</spatialType>
        <dataType>scalar</dataType>
      </variable>
    </interfacePort>
    ...
  </interface>
  <initialisation> ... </initialisation>
  <dynamic> ... </dynamic>
</modelComponent>
```

Figure 3: XML code snippet describing the structural and behavioural characteristics of a model component with one interface port in more detail. Additional meta–information like component descriptions can be included as well.

before the model is executed.

The model developer specifies dependencies between model components by relating the input port of a model component requiring a variable to the output port of a component providing the variable. A link between two components for a specific variable is therefore given by:

```
link = (interfacePort1, interfacePort2)
```

After the links between model components are defined by the model developer, the modelling framework needs to evaluate all links to ensure a sound structural setup of a coupled model. In a first step, the availability and direction of both interface ports need to be assessed for each link, that is, the given variables must be present in the components and the directions must match output and input. In a second step, properties of input and output of a particular variable need to match. Depending on the used variable, this evaluation can be executed for example on the level of fundamental data types provided by programming languages such as float or Boolean. Additional semantic information is incorporated into the evaluation step such as extent and grid discretisation for raster–based spatial data types [Wesseling et al., 1996].

The evaluation steps need to be performed for all links occurring between model components. In case that all links are valid and all input requirements of the model components are fulfilled the coupled model can be executed. Otherwise, the modelling framework informs the model developer that the coupled model cannot be executed properly.

3 SOFTWARE PROTOTYPE IMPLEMENTATION

The previous section introduced a formalised description of model components that allows assembling and structural evaluation of coupled models. Here, we outline how the semantic information of these structural and behavioural characteristics are represented in platform–independent XML format. As an example, we consider a simple model component simulating soil water storage. The input to the component is a spatial variable with

```
class Storage(ModelComponent):
 """simple storage calculation """
 def _init_(self):
  self.storage = scalar(0)
  self.fraction = scalar(0.2)
  self.precipitation = scalar(0)
  self.setInput("precipitation")
  self.setParameter("fraction")
  ModelComponent._init_(datetime(2000,1,1),datetime(2010,1,1),\
                                                    timedelta(days=1))
 def process(self):
  self.storage = self.storage + self.precipitation
  self.storage = self.storage - self.fraction * self.storage
```

Figure 4: Code snippet for the storage model component. The model developer can describe structural and behavioural characteristics in a declarative manner. Note that all arithmetic operations are performed on spatial raster data types.

precipitation values. The storage is calculated as accumulated precipitation diminished by an outflow fraction. The fraction is a component parameter, the storage is the component output.

Figure 3 shows an excerpt of the model component description. The characteristics of the model component are mapped into a hierarchical XML structure. For brevity, the initialisation describing parameters and state variables, the temporal characteristics and the output port are omitted. As component interface, one port is representing the input variable. The variable precipitation contains further specification about spatial discretisation and type information.

The preparation of the semantic XML format remains tedious for environmental scientists with basic programming knowledge. We intend to minimise the technical complexity and intrusiveness of a semantic layer in a declarative development process and therefore use the Python scripting language [c.f., Karssenberg et al., 2010; Schmitz et al., 2012] in combination with its introspection capabilities to obtain the component descriptions. Introspection is the functionality of programming environments to gather information, to access and modify objects at runtime allowing the modelling framework to query model components about specific properties. Introspection support is also provided by other programming languages like Java and C#, and facilitated within modelling frameworks such as OpenMI [Moore and Tindall, 2005] or the ICMS [Rahman et al., 2004] as well.

Figure 4 shows a potential implementation of the storage model component that enables the framework to generate the XML description shown in Figure 3. The framework base classes provide templates guiding the model developer in the implementation of required sections such as component initialisation. The specification of parameters, state variables and component interface is completed with framework methods such as setInput or setParameter. The dynamic behaviour of the component is given by a 10–year time span with a daily time step. The storage calculation itself is implemented in the process method. If the model developer modifies the component interface, the XML is updated.

4 Conclusion and future work

The development of integrated models involves the construction of individual model components with domain specific functionality, and the coupling of various model com-

ponents with complementary functionality to larger systems. The presented semantic layer enhances a modelling framework designed for the exploratory development of spatio–temporal models. The formal descriptions of component interfaces and interactions provide guidelines for component–based development practices, lead to a better interoperability within the modelling framework, and aid and protect the domain specialist in a sound setup of integrated models.

The properties of component interfaces and variables can be expressed in machine readable XML format. By providing templates and framework methods in the Python scripting language, these XML descriptions can be derived without excessively interfering in a declarative component model development process. However, adding semantic information to user defined variables still needs to be done by the model developer explicitly. This requires discipline as part of the good modelling practises and cannot be fully enforced by a modelling framework.

Our current work is on extending the semantic representation of coupled models with hierarchical levels, including the formal description of different spatial and temporal discretisations present in the subsystems. We use the resulting semantic description to complete model application tasks such as execution and calibration. Future work is on exploring how the semantic layer can be used to establish interoperability with components not developed in the framework such as is the case for components residing as web service, developed in other programming languages or modelling frameworks.

5 Acknowledgements

The Flemish Institute for Technological Research (VITO) provided financial support.

References

Argent, R. M., J.-M. Perraud, J. M. Rahman, R. B. Grayson, and G. M. Podger. A new approach to water quality modelling and environmental decision support systems. *Environmental Modelling & Software*, 24(7):809–818, 2009.

Argent, R. M. An overview of model integration for environmental applications–components, frameworks and semantics. *Environmental Modelling & Software*, 19 (3):219–234, 2004.

Athanasis, N., K. Kalabokidis, M. Vaitis, and N. Soulakellis. The Emerge of Semantic Geoportals. In *On the Move to Meaningful Internet Systems 2005: OTM Workshops*, pages 1127–1136. Springer Berlin / Heidelberg, 2005.

Baglioni, M., E. Giovannetti, M. V. Masserotti, C. Renso, and L. Spinsanti. Ontology-supported Querying of Geographical Databases. *Transactions in GIS*, 12:31–44, 2008.

Beck, H., K. Morgan, Y. Jung, S. Grunwald, H. young Kwon, and J. Wu. Ontology-based simulation in agricultural systems modeling. *Agricultural Systems*, 103(7):463–477, 2010.

Beck, M. B., A. J. Jakeman, and M. J. McAleer. Construction and evaluation of models of environmental systems. In Beck, M. B., A. J. Jakeman, and M. J. McAleer, editors, *Modelling change in environmental systems*, pages 3–35. John Wiley & Sons Ltd., New York, 1993.

Best, B. D., P. N. Halpin, E. Fujioka, A. J. Read, S. S. Qian, L. J. Hazen, and R. S. Schick. Geospatial web services within a scientific workflow: Predicting marine mammal habitats in a dynamic environment. *Ecological Informatics*, 2(3):210–223, 2007.

Booch, G., R. A. Maksimchuk, M. W. Engel, B. J. Young, J. Conallen, and K. A. Houston. *Object Oriented Analysis and Design with Applications*. Addison Wesley, 2007.

Buccella, A., A. Cechich, and P. Fillottrani. Ontology-driven geographic information

integration: A survey of current approaches. *Computers & Geosciences*, 35(4):710–723, 2009.

de Kok, J.-L., G. Engelen, and J. Maes. Towards model component reuse for the design of simulation models a case study for ICZM. In Swayne, D. A., W. Yang, A. A. Voinov, A. Rizzoli, and T. Filatova, editors, *Proceedings of the iEMSs Firth Biennial Meeting: International Congress on Environmental Modelling and Software (iEMSs 2010)*, pages 1215–1222. International Environmental Modelling and Software Society, Ottawa, Canada, 2010.

Galton, A. and R. Mizoguchi. The water falls but the waterfall does not fall: New perspectives on objects, processes and events. *Applied Ontology*, 4(2):71–107, 2009.

GDAL. Geospatial Data Abstraction Library, 2009.

Gruber, T. R. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.

Gruninger, M. The ontological stance for a manufacturing scenario. *Journal of Cases on Information Technology*, 11(4):1–25, 2009.

Hinkel, J. The PIAM approach to modular integrated assessment modelling. *Environmental Modelling & Software*, 24(6):739–748, 2009.

Hofmann, M., J. Palii, and G. Mihelcic. Epistemic and normative aspects of ontologies in modelling and simulation. *Journal of Simulation*, 5(3):135–146, 2011.

Jakeman, A. J., R. A. Letcher, and J. P. Norton. Ten iterative steps in development and evaluation of environmental models. *Environmental Modelling & Software*, 21(5): 602–614, 2006.

Janssen, S., I. N. Athanasiadis, I. Bezlepkina, R. Knapen, H. Li, I. P. Domínguez, A. E. Rizzoli, and M. K. van Ittersum. Linking models for assessing agricultural land use change. *Computers and Electronics in Agriculture*, 76(2):148–160, 2011.

Karssenberg, D. and K. de Jong. Dynamic environmental modelling in GIS: 1. Modelling in three spatial dimensions. *International Journal of Geographical Information Science*, 19(5):559–579, 2005.

Karssenberg, D., O. Schmitz, P. Salamon, K. de Jong, and M. F. P. Bierkens. A software framework for construction of process-based stochastic spatio-temporal models and data assimilation. *Environmental Modelling & Software*, 25(4):489–502, 2010.

Letcher, R. A., B. F. W. Croke, and A. J. Jakeman. Integrated assessment modelling for water resource allocation and management: A generalised conceptual framework. *Environmental Modelling & Software*, 22(5):733–742, 2007.

Liu, J., C. Peng, Q. Dang, M. Apps, and H. Jiang. A component object model strategy for reusing ecosystem models. *Computers and Electronics in Agriculture*, 35(1):17–33, 2002.

Lutz, M. and E. Klien. Ontology-based retrieval of geographic information. *International Journal of Geographical Information Science*, 20(3):233–260, 2006.

Madin, J., S. Bowers, M. Schildhauer, S. Krivov, D. Pennington, and F. Villa. An ontology for describing and synthesizing ecological observation data. *Ecological Informatics*, 2 (3):279–296, 2007.

Mark, D. M., A. G. Turk, and D. Stea. Progress on Yindjibarndi Ethnophysiography. *Lecture Notes in Computer Science*, 4736 LNCS:1–19, 2007.

Moore, R. V. and C. I. Tindall. An overview of the open modelling interface and environment (the OpenMI). *Environmental Science & Policy*, 8(3):279–286, 2005.

OWL. OWL Web Ontology Language Guide, 2004.

Pullar, D. Simulation Modelling Applied To Runoff Modelling Using MapScript. *Transactions in GIS*, 7(2):267–283, 2003.

Rahman, J. M., S. P. Seaton, and S. M. Cuddy. Making frameworks more useable: using model introspection and metadata to develop model processing tools. *Environmental Modelling & Software*, 19(3):275–284, 2004.

Refsgaard, J. C., J. P. van der Sluijs, A. L. Højberg, and P. A. Vanrolleghem. Uncertainty in the environmental modelling process - A framework and guidance. *Environmental Modelling & Software*, 22(11):1543–1556, 2007.

Rizzoli, A. E., M. Donatelli, I. N. Athanasiadis, F. Villa, and D. Huber. Semantic links in integrated modelling frameworks. *Mathematics and Computers in Simulation*, 78(2–3): 412–423, 2008.

Saiful Islam, A. and M. Piasecki. A generic metadata description for hydrodynamic model data. *Journal of Hydroinformatics*, 8(2):141–148, 2006.

Schmitz, O., D. Karssenberg, K. de Jong, J.-L. de Kok, and S. M. de Jong. Integrated ecosystem modelling: a software environment for component construction and coupling. 2012. In prep.

Scholten, H., A. Kassahun, J. C. Refsgaard, T. Kargas, C. Gavardinas, and A. J. Beulens. A methodology to support multidisciplinary model-based water management. *Environmental Modelling & Software*, 22(5):743–759, 2007.

Szyperski, C. *Component Software: Beyond Object-Oriented Programming*. Addison Wesley, 2 edition, 2002.

Uschold, M. and M. Gruninger. Ontologies: Principles, methods and applications. *Knowledge Engineering Review*, 11(2):93–136, 1996.

van Deursen, W. P. A. *Geographical Information Systems and Dynamic Models*. Koninklijk Nederlands Aardrijkskundig Genootschap/Faculteit Ruimtelijke Wetenschappen, Universiteit Utrecht, Utrecht, 1995.

Villa, F. A semantic framework and software design to enable the transparent integration, reorganization and discovery of natural systems knowledge. *Journal of Intelligent Information Systems*, 29(1):79–96, 2007.

Villa, F. Integrating modelling architecture: a declarative framework for multi-paradigm, multi-scale ecological modelling. *Ecological Modelling*, 137(1):23–42, 2001.

Villa, F., I. N. Athanasiadis, and A. E. Rizzoli. Modelling with knowledge: A review of emerging semantic approaches to environmental modelling. *Environmental Modelling & Software*, 24(5):577–587, 2009.

Voinov, A., C. Fitz, R. Boumans, and R. Costanza. Modular ecosystem modeling. *Environmental Modelling & Software*, 19(3):285–304, 2004.

Wesseling, C. G., D. Karssenberg, W. P. A. van Deursen, and P. A. Burrough. Integrating dynamic environmental models in GIS: the development of a Dynamic Modelling language. *Transactions in GIS*, 1:40–48, 1996.