

A semantic framework and software design to enable the transparent  
integration, reorganization and discovery of natural systems  
knowledge.

**Ferdinando Villa**

Ecoinformatics Collaboratory  
Gund Institute for Ecological Economics and Department of Botany  
University of Vermont

590 Main Street  
Burlington, VT, 05405  
[U.S.A.](#)  
ferdinando.villa@uvm.edu

## Abstract

I present a comprehensive conceptualization that unifies representation and integration of natural data while at the same time providing a workable computational framework for developing and running simulation models. The approach is based on a long-standing principle of scientific investigation: the separation of the ontological character of the object of study from the semantics of the *observation context*, the latter including location in space, time, classification schemata, and any other observation-related aspects. I will show how the object-oriented paradigm allows an efficient implementation of the concept through the abstract model of a *domain* class, which relates to the idea of *aspect* in software engineering.

This conceptualization allows us to factor out two fundamental causes of complexity and awkwardness in the representation of knowledge about natural system: (a) the distinction between data and models, which are both seen here as generic *modules* or information sources; (b) the multiplicity of states in data sources, which is handled through the hierarchical composition of independently defined domain objects, each accounting for all states in one well-known observational dimension, such as space or time. This simplification leaves modelers free to work with the bare conceptual bones of the problem, encapsulating away complexities connected to data format, scale, and the fact that data may either come from known data sources or be obtained by simulation. I will show how a software system, based on established representational frameworks, can be efficiently designed to allow an operational implementation of the approach, referring to

explicit ontologies to unambiguously categorize the semantics of the objects of study, and allowing the independent definition of a global *observation context* that users or automated retrieval systems can redefine as required by the needs of the investigation. I will briefly discuss an existing software prototype and its applications to multi-scale, multi-paradigm modeling, intelligent database design, and web-based collaboration.

## Introduction

Computer science has proved its great effectiveness in providing users with operational metaphors that have made many complex tasks intuitive. Nevertheless, scientific investigation still relies largely on the manipulation of raw data. Despite the powerful capabilities of dedicated analytical software such as GIS and statistical packages, we have not yet identified a framework that allows natural system scientists to work with natural world representations (data and models) using the same language that we use to think about natural phenomena and entities. Long-established achievements in personal productivity, information management, and scientific software allow us to work efficiently with spreadsheets, GIS coverages, differential equations; still, such activities use a very low level of abstraction compared to our thinking in terms of biomass, temperature, or species diversity. Complexities related to data format, scale, different granularities in space and time, and ambiguous, often opaque semantics make the workflow of natural systems investigation slow and awkward, severely limiting a scientist's creativity and productivity.

Identifying synthetic approaches to enable more productive ecological knowledge representation and management is ultimately a technical problem, but it cannot be addressed at the technical level before a breakthrough in language is reached. In this paper I will propose a conceptual framework that unifies and simplifies access to and

integration of data and models while at the same time providing a sound computational blackboard for simulation modeling. The framework I present builds on the most recent efforts in knowledge engineering applied to ecological research to address the following priorities:

- Give scientists the ability of searching and retrieving generalized knowledge sources (data and models) in a predefined observation context, leaving issues of scale, unit translation and enforcement of compatibility, and measurement of accuracy to an automated system.
- Enable arbitrary recombination and composition of models and analytical pipelines based solely on compatibility of semantics.
- Enable a network-enabled collaboration model where data, models, and modeling scenarios can be communicated, modified and exchanged in the same ways that text-based, discursive knowledge is.

The conceptualization presented here strives to “generalize away” two fundamental causes of complexity in natural system representation:

1. the distinction between data and models, seen as special cases of a generic “module” or information source;
2. the multiplicity of states in data sources, defined and managed through the hierarchical composition of independently defined objects each of which determines a multiplicity of states in one well-known representational dimension.

Such a simplification leaves us with the basic concept to work with and eliminates complexities relating to data format, scale, and nature of the information. The concept

presented here differs from current model integration frameworks (e.g., ANL, URL; DMSO, URL; Fishwick, 1996; Mosterman and Vangheluwe, 2000; Villa and Costanza, 2000; Vangheluwe et al., 2001), toolkits, and languages (e.g., Minar et al., 1996; Fritzson and Engelson, 1998) by advocating a representational breakthrough before dealing with the engineering aspects of enabling the data flow between distributed and heterogeneous sources and analytical processes. After a description of the conceptual framework, I will show how a software system can be efficiently designed to allow an operational translation of this concepts that is by its own nature distributed and web-enabled.

## **Approach**

The framework I propose in this paper extends previous work on integration infrastructure (Villa and Costanza, 2000; Villa, 2001) and is based on the fundamental epistemological separation between an object of study and the process of its observation. Each object of study can be associated to a *concept* and an *observation context*. The concept is a formalization of the ontological nature of the object, and is immutable by definition. The observation context defines all the aspects of an “observation window” such as temporal and spatial extent and granularity, the classification schemata employed, the authors that have reported the data, etc. The observation context defines a set of constraints whose representation is not necessarily and not entirely fixed: representational aspects connected to scaling, such as granularity and extent, can be manipulated by the observer without modifying the nature (semantics) of the object. I argue that reflecting this fundamental epistemological distinction in the representational model can be

instrumental in developing a more efficient semantics for natural system representation.

### ***Formalizing the semantics of the object of study***

Long-standing research in knowledge engineering has only recently started to percolate down to the natural sciences. With a renewed attention to semantics issues and naming problems, efforts that try to match the concept incorporated in an object of study to a formally specified *semantic type* are now becoming common, with the most activity occurring in the field of taxonomy and biodiversity conservation (Stevenson et al., 2003; GBIF, URL; TDWG, URL). In most of these studies, the semantic type is a pointer into an explicit ontology, where the knowledge is mapped to the terms of a controlled vocabulary and, in the best cases, a set of attributes and relationships that link the terms of the vocabulary into a conceptual structure. As an example, Ludaescher et al. (2000) define a semantic map for a set of problems in neural transmission, and use it to query heterogeneous repositories of conceptually related data sources.

A semantic map is a structured equivalent of the set of keywords traditionally used to characterize the semantics of stored datasets. Compared to the rather primitive keyword approach, a semantic map can also serve as a precious aid in browsing and querying of knowledge bases, as it connects concepts in a meaningful way and lends itself to powerful and intuitive forms of interactive graphical representation. Linked as it is to disciplines and their interpretation, there can hardly be a requirement of uniqueness in the semantic typing of an object: in most applications, an object can refer to several semantic maps, but can only have one position in each.

For the purposes of this discussion, objects that share the same semantic type are representations of the same concept. As a result they can be observed in the same way. The semantic type defines unambiguously the aggregation properties of the object, the elementary unit of measurement that can be applied to it, and the storage properties of any measurement of its state.

## **Semantics and role of the observation context**

Natural system sciences, and Ecology in particular, have long been concerned with issues of scale, and with the effect of the scale of observation on the perceived properties of a system. It is easy to understand the fundamental role of scaling issue when considering that the observers are also the components of the systems under study, as is the case in highly integrative Ecology. The realization that scaling is rooted in observation, as opposed to the inherent semantics of the object of study, has been prominent, if often implicit, in ecological literature for at least two decades (Allen and Starr, 1982).

Separating the semantics of the object of study from the aspects that pertain to its observation can be the starting point of a new integrative synthesis in ecological knowledge representation. To enable this, it is necessary to make a fundamental simplifying assumption: that the observation context can be separated into orthogonal axes, each accounting for a domain of observation that has well-defined semantics, methods, and interfaces.

In most cases, the multiplicity of states in data can be attributed to the observation context, not to the semantic type. For example, timeseries data detail the various states of

a concept in a certain extent of time, viewed at a defined granularity. GIS coverages specify different states in different portions of a given spatial extent. By encapsulating the observation context of an object into separately characterized observational dimensions, we can design infrastructure that allows us to only use the semantic types to assemble models and analytical procedures, concentrating on the bare conceptual aspects and leaving the task of merging of the observation contexts of heterogeneous knowledge sources to an automated mediating engine. The separation of the various axes of observation into independently defined “observation domains” - such as space or time - is key to the possibility of implementing such a mechanism in a modular and flexible way.

The observation context is as relevant to the analytical phase of the scientific workflow as it is to the process of searching and retrieval knowledge from repositories. In fact, predefining a semantic type and an observation “window” is equivalent to defining a query on a knowledge base. The implications of using formally specified semantic maps to perform queries (possibly by translating natural language interrogations or interactively browsing a graphical knowledge map) have been extensively explored elsewhere (Ludaescher et al., 2000, and references therein). Once a concept is selected, an user can look for all objects that express that concept in a particular setting of time, space, classification, or other observational domain. The retrieved objects can be returned unmodified or rescaled to fit a user-specified extent or granularity. This applies intuitively to space and time, but may apply to less obvious domains such as bibliographic source. As an example, if a knowledge base contains the same concept measured at the same time and space by different authors, a query may return both

measures and their aggregated (average) measure if the granularity and extent of the bibliographic source domain is not specified – i.e, no specific source was requested. This would depend on having incorporated in the representational semantics of the repository a formal definition of a bibliographic “dimension” of the observation context, whose methods would be isomorphic with those used to aggregate over space and time.

In this framework, *data* can be thought of as the simplest representation of an object of study, in which the measurements of a concept in a given observation context are known directly. A system represented by data has a “natural” observation context, that is conventionally captured in the associated metadata. Its state can be observed in a different context by transforming it (e.g. by aggregating, distributing or subsetting) over the represented extents, or extrapolating it to a different extent.

A system represented by algorithmic knowledge (model) may be defined more or less generally, from a completely abstract specification to one that depends on a particular time, space, and scale. To access its states in one of the permissible observation contexts, they need to be calculated.

### **Representation of models and analytical processes**

Models, where the information necessary to produce the states is defined algorithmically, can be expressed as a hierarchy of generic knowledge sources (modules) connected in a dependency chain, where each module has a semantic type and can adopt a specified observation window. The structural properties of the model do not depend on time, space or other observation-related aspect; scheduling, representation, mode of visualization depend on the semantics and nature of the composite observation context implicit in the

model and the data used as input.

Before states can be calculated across the model hierarchy, the observation contexts of the components must be compounded into an overall context, their compatibility assessed, and a strategy (composed of a set of transformations) defined to produce the states. I will detail a software design implementing this concept in the following sections.

Building a model based on dependencies, using a formally specified semantic type to guide the matching of inputs and outputs is a major simplification over the common box-and-arrow representational paradigm adopted in many modeling toolkits (e.g., STELLA: HPS, 1995; Costanza et al., 1998) where the connections usually imply a temporal sequence or control. The separation of the observation context makes much control flow (particularly looping over multiple states) unnecessary, and allows a system to automatically determine the proper sequence of calculation of its states, as well as the potential for parallel calculation. Box-and-arrow dependency diagrams can be much more easily designed by non-technical users, particularly with if explicit semantic types aid the choosing and matching of connections.

The uniformity of representation has many advantages. Models can represent analytical steps (such as GIS processing, statistical analysis or function optimization) just as well as interpretive or predictive models of natural phenomena. By virtue of semantic type composition, analysis steps acquire a semantics of their own and can become legitimate components of other models. As an example, a GIS processing step may calculate the fractal dimension of a surface. By representing the calculation step as a module and its result as its state, the observational context of the original data will be inherited by the

fractal dimension data, which will maintain the temporal and spatial extents of the input map. At this point, any further processing that uses the result as input will be able to use these informations to enforce proper design and correct use of the information when coupled with other spatially- and temporally- explicit data and models. The “metadata” in the resulting module will be in large part created automatically by inheriting the observation context.

The observation context encapsulates much of the complexity that scientists have to deal with when manipulating data and models, without getting in the way of conceptualizing and analyzing. By separating the semantics of the observation into clean abstraction that do not influence the semantic type, we can model natural systems using the same intuitive language we use when conceptualizing.

Another strength of this approach is that a model can be defined without any reference to a particular observational dimension, e.g. space, and used with data that contain it. The calculation of, e.g., a difference equation model using spatially explicit data as input would propagate the spatial nature of the input data across the model hierarchy and make the model in fact spatial, without user intervention. This mechanism is detailed in the following, where I will sketch an object-oriented interpretation of the concepts expressed this far, meant to illustrate the translation of the approach into an operational software design.

## **An object-oriented design: the Integrating Modelling Architecture.**

The most general representation of a system in an object-oriented framework can be

thought of as a set of interconnected *modules*, each corresponding to a precisely identified object of study. This section deals with the possible types of modules, their properties and mutual relationships, and the operational translation of the semantic type and observation context. The principles exposed here are incorporated in the most recent design of the Integrating Modelling Architecture (IMA: Villa, 2001; Villa, URL), an open source, XML-based integrative architecture for natural system modeling. In this section I concentrate on the conceptual design of the system. A more detailed, if brief, description of the implementation is given in a later section of this paper.

### **Modules as generic knowledge sources.**

A *module* is an object that represents an entity in the natural world; its semantics is precisely defined by a semantic type, i.e., a pointer or path into an externally specified ontology. In the IMA, modules are specified by documents in the XML language that refer to a formal schema (Villa, 2001). Related to its semantic characterization, each module can have a state which belongs to a precisely identified storage type, such as integer, floating point number, string, etc. Most modules of interest to natural system modeling will contain numeric states. To be fully specified, the state must be given along with the actual units of measurement employed, which must be compatible with the unit class implied by the semantic type. The semantics is also the key to its aggregation behavior: as an example, properties like temperature have different aggregation behavior than quantities like biomass, in that they must be averaged, rather than summed, when redistributed or aggregated over space or time.

The generic module is an abstract specification that fits both data- and algorithmically-specified knowledge sources. Data modules come with predefined state(s) and observation context. Algorithmically defined modules have additional specification syntax that allows the algorithms and their dependencies on the state of other modules to be precisely characterized, and a dependency graph to be constructed and used at the time of calculation.

Modules in a dependency graph are substitutable based on the semantic type. The compatibility of storage type and elementary units of measurement is guaranteed by our definition of semantic type. Thus, a model that uses grass biomass and temperature to determine the increase in body weight in an organism can take the temperature indifferently from a timeseries or a complex global change model, as long as the observation contexts are compatible.

While the state of a module can have multiple values in a given observation context, a module's state is conceptually always atomic: it does not hold information about its multiplicity, as this is not a stable property of the object, being entirely dependent on the observation context. If initial data are available, they're merely used to define a default state when the object is observed through its "native" observation window. No concept of array or matrix is necessary, as it does not correspond to an observational characteristic of the states of natural entity.

On the other hand, a containment relationship is defined for modules and has a precise meaning, which depends on the semantic type. A set of modules may result as the effect of a containment relationship, and it is legal in the IMA as long as the set of modules is

itself a module. In this case, the state of a module that contains other modules may be defined, according to its semantic type, as the union or the aggregation of the states of its components. As an example, individual-based models can contain modules that represent different properties such as body weight, age, temperature (Figure 1): such properties have a well defined semantic type of their own, but they only make sense in the context of the individual. It must be noted that in such cases, the contained modules cannot exist alone: in other words, the semantic of the submodules is defined “hierarchically” in terms of their relationship with the containing module. The containment relationship requires a specialized set of attributes in a traditional ontology framework in order to accommodate such concepts.

Another possible containment relationship results from the aggregation of similar components. For example, a population can be defined in terms of the individuals that compose it. In this case, the individuals' states can be automatically aggregated into the population's (e.g., the total biomass will be the sum of individual body mass), and additional semantics can be given to account for properties that only appear as a result of the aggregation (emergent properties), such as the age structure.

Modules can be linked together in a dependency relationship, either explicitly or by using a heuristic process based on semantic type and label. In the IMA, a containment relationship can be used to define a subset of modules that are searched when resolving dependencies, without having to specify links directly. Users are notified and asked for input only when ambiguous links are possible. When states are calculated, the linked modules are sorted in topological order according to the structure of dependencies. The

definition of a strategy to calculate the states across the whole model hierarchy depends at this point on how the observation context is defined and implemented.

### **Domains: an object-oriented translation of the observation context**

The observation context is represented in each module as a set of independently defined objects called *domains*, that can only exist within modules. Each domain accounts for one “axis” of the observation context: e.g., space or time. Each domain specifies all the necessary information about the corresponding concept in the semantics of the observation: e.g., in a “discrete time” domain the specification syntax requires a start time, an end time, and the duration of a step. Modules *adopt* domains from their original specifications or by inheriting them from the modules they are linked to in a dependency or containment relationship. Adopting a domain with a single step (multiplicity = 1) defines the extent of existence in that domain: e.g. the time period where a data object has the stored value. Adopting a domain that specifies more than one time step determines a multiplicity of states in time for the module, and should be matched by appropriately defined input data. At the same time, the module may adopt a space domain defined through a set of polygons or raster cells covering a specified extent, and possibly loaded from a GIS coverage. The full state of each module is the Cartesian product of the set of states implied by each domain, and the total multiplicity of its state is the product of the multiplicities attributable to each domain.

The set of all domains adopted is the module's observation context. In an object-oriented implementation, domains are polymorphic, and their associated methods can calculate,

merge, morph across different representations of the same domain (e.g. rasters, polygons) using methods and techniques tailored to the specific nature of the domain and its representation. A domain's abstract model is defined generally and precisely in terms of granularity and extent, and new classes of domains can be defined by deriving representations of the granularity and extent that fit the domain, and providing an interface to them in the terms required by the abstract specification.

Most sources of multiplicity in a concept's state can be encapsulated into domains. For example, a bibliographic source domain can be devised that gives an object different states according to the source or author that has published their value. In this case, the granularity is the number of different sources that contain different values of the object, and the extent is the set of sources.

In the example of Figure 1, a population module contains a number of individual modules. By virtue of the containment relationship, the body mass of the individual aggregates into the population biomass automatically, since the population class is derived from the individual class, and the biomass semantic type knows the proper aggregation type for a containment relationship. The individual growth depends on nutrient availability and temperature, which are linked from external modules. The individual model does not adopt domains, but time and space are inherited from the first module in the dependency chain (e.g., temperature) and compounded with all others. This way, the compatibility of the time and space domains adopted by the individuals is assessed through the individual class. The containment relationship makes the population adopt the common observation context of the individuals, that in turn comes from the

compounded nutrient data and climate change model. Using data for temperature and a model for nutrients would not change the mechanism or the interface for a user who builds the model.

This use of domain objects for mediation of semantics and methods is conceptually related to the field of aspect-oriented programming (Grundy, 2000), a technique that allows to modularize cross-cutting aspects of a system. Nevertheless, the IMA ideas did not evolve with aspect-oriented programming in mind and the system does not employ the technique in a software engineering sense.

### ***Operations on a module hierarchy***

To be useful, an implementation must be able to produce all states of a concept in any admissible observation context. This is accomplished in the IMA by two fundamental operations defined on the hierarchy, which are the main functionalities of the system and the key to all uses of the conceptualization presented here.

- **Contextualization**, where an overall observation context is defined, either by the user or as the compounded observation context of all the modules; during this phase, the compatibility between linked modules is assessed, and a high-level transformation and calculation strategy to produce the states is defined;
- **Domainization**, where the high-level strategy is passed to a software engine that carries it on and produces the states of all modules in the global observation context.

The states resulting from domainization can be visualized according to the semantics of the observation context. The rest of this section explains the two operations and how

module and domains are used to carry them on.

## Contextualization

The overall observation context over a hierarchy of modules is defined by compounding the domains adopted by the modules in bottom-up order across the chain of dependencies and the containment structure of the model hierarchy. A module is exposed to all the domains of all modules it depends on or contains. Different actions are taken according to whether the compounding module originally adopts the domain or not. If module  $M1$  depends on module  $M2$ , and module  $M2$  adopts domain  $d$ , the two possible actions are:

- if  $M1$  does not adopt domain  $d$  from its original specifications, it adopts a copy of domain  $d$  from the linked module. For example, a non-spatial equation becomes spatial by adopting a spatial operand; its state will be replicated to cover the whole multiplicity of  $d$ , and the equation will be calculated the same number of times with all the different values of  $M2$  in  $d$ , in the order specified by ordering rules specific to domain  $d$ 's class.
- If  $M1$  adopts  $d$  from its original specifications, the domain in  $M1$  is *compounded* with the domain of the same class in  $M2$ . The operation of compounding involves calls to domain  $d$ 's interface methods to assess the compatibility of extent and granularity between the two domains, and if compatible, the generation of a transformation that can compute a minimum common denominator representation. The transformation, if any, is memorized as part of the overall link strategy, and the common representation is memorized in  $d$  and carried over to the next level.

If module *M1* adopts *d* and *M2* does not, *M2* is considered domain-neutral with respect to *d*, and its state will be used untransformed (with the possible exception of unit conversion) to calculate all states of *M1*.

If the contextualization process fails because of domain incompatibility, an error is generated and the operation is interrupted. If the operation is successful, it produces the overall observation context of the model hierarchy, and a strategy represented by a set of transformations that, carried out by an appropriate calculation engine, will “filter” the states of the dependent modules and reduce them to the same observation context of the requiring ones.

After the bottom-up domain compounding phase, a second pass is done to expose the whole hierarchy to the newly calculated overall context, this time top-down. This step has the purpose of defining partially defined domains using the overall context to provide defaults. It is common that algorithmically defined modules are defined very generally, without a precise statement of extents and granularities (although they may contain specific extents in order to prevent application to situations where they have been proved not to apply). In such cases, the top-down inheritance of domain properties has the role of defining the missing information, thus preparing the modules for calculation.

The compounded context in the root module is the overall view of all the axis of the observation context across the hierarchy, and is defined as the *natural context* of the model. The same process that is used during linking to contextualize a model in the context of another can be used to impose a user-defined observation context. When the overall context is defined, all modules know precisely their state storage requirements,

and the system knows the transformation strategy and the precise ordering in which the states should be calculated. The contextualization of a single data source produces an empty strategy.

In a database context, the contextualization operation assumes an important role in guiding the query process and allowing automatic enforcement of correct model design. A user can download a generic module with predefined dependencies, e.g., the algorithm that will calculate the predator-prey dynamics between populations of two species. To be representative of a real-life situation, the populations should occupy the same area and exist at the same time. The model will contain default no-op values as retrieved, but the user will be able to point to a dependency and look for modules that match its semantic type and the current overall observation context. By definition, only modules that have the same semantic type will be admitted to satisfy the dependencies, and all modules that do will be valid. When the first population data module is retrieved and linked to the algorithm module, the observation domains adopted by the population (e.g. time and geographic space) will trigger recontextualization, which will define the overall context of the model being built to match that of the population. The second population module will be searched for in the current overall observation context: unless the constraint is explicitly loosened by the user, no population that does not share the same space and time as the one retrieved first will appear as a result of the next query. This mechanism automatically defines and enforces a discipline in creating meaningful models. In general, as new modules are inserted, the overall context is incrementally narrowed, which automatically restricts the search range for the ones that follow. The

exact nature of the modules is not an issue: as long as the observation contexts are compatible and the semantic type is correctly matched, data and dynamic models of different nature can be freely mixed.

New domains can be devised that are capable of determining and enforcing compatibility of more sophisticated aspects. For example it is possible, although hard to do at the present state of knowledge, to design an ecological domain that will enable the selection of only prey species that are known to be preyed upon by a specific predator once the predator is selected. The concept is general enough to accommodate most design problems. The modular nature of the IMA allows users to develop and add support for new domains in an uncoordinated, flexible way.

## **Domainization**

If the process of contextualizing a model hierarchy produces the overall observation context and a strategy to produce the modules' states, the process of *domainization* produces the states that are implied by the overall context. This may require calculating states when the knowledge base is described algorithmically, and applying scaling transformations when the same domain is present at different granularities or extents throughout the hierarchy of modules. Domainization, the equivalent of running a model in the conventional sense, starts with a topological sort of the dependency graph (assessing the potential for parallelization) and calculates states for all modules, in dependency order, along the finest granularity of each domain in the overall observation

context. The order of calculation is chosen accordingly to domain-specific rules, as some domains such as time imply an irreversible sequence of states while others (such as space) do not implicitly require that, but may have internal dependencies (e.g. when representing flows in space) that require a specific ordering of calculation of different states within the domain.

Separating the contextualization and domainization process allows a clean design where the domainization strategy is calculated independently and expressed in a high-level representation (e.g. in XML) which is later fed to a “domainization engine” that is capable of carrying it on optimally, using distributed parallel processing when possible and available. Web services provide a promising platform to enable a clean separation of the different processing stages components of such a system.

The high-level representation of the transformation strategy is also suitable to independent optimization, and can be stored or cached as necessary to speed repeated calculations in the same observation context. An important property of a well-defined transformation strategy is the ability of estimating in advance the loss of accuracy that the domainization process will cause compared to the use of untransformed data. This feature is extremely important in any integration architecture, since error propagation is even more of an issue when the output of an analytical process can be readily used as input for another, causing potentially infinite propagation of error. Estimates of accuracy can be permanently carried in the resulting modules' data model and influence the confidence that users should put in the results.

In the IMA, the domainization process produces a separate module hierarchy whose

modules contain all states implied by the context, and a copy of all the domains after the contextualization step, but no algorithm specifications are left in the modules. In other words, the result is a hierarchy composed of data modules only, where the dependency relationship in the original module is substituted by containment so that the original structure is still recognizable. These “result modules”, represented in XML just like the original model hierarchy, can be independently visualized and saved to permanent storage.

## **Visualization**

While the IMA does not mandate any standard for visualization, the observation context of each module is usually sufficient to determine the proper way to visualize the state of each module. For example, the state of a module with multiple states in both time and space is best represented as an animation of maps, while the state of a module that only has multiple numeric states in time is typically represented as a line graph. The visualization engine uses the popular MIME types used by browsers and user interfaces to map a set of states in a particular context to a given MIME type, and invokes a third party plug-in, if present, to generate the corresponding representation (possibly handled through an external visualization program).

## **Implementation principles**

An open-source prototype implementation of the principles illustrated above, the Integrating Modelling Toolkit (IMT), is available on the World Wide Web (Villa, URL).

Despite its prototype stage, the IMT is being actively used in database and modeling projects. While it is impossible to fully illustrate its design in this paper, I will sketch its salient features in this section.

The IMT is an extensible object system that uses the XML language to define and serialize all of its components. The operational units (modules) and the domain objects are represented in XML according to XML-specified “grammars”, schema documents that are used to validate the module specifications and creating the schemata for permanent storage of modules. Modules belong to classes that support the common inheritance and overloading mechanisms. New classes can be added to the system by writing the corresponding XML grammars and, if necessary, extension code in the native C++ language or in any language that can be linked to it.

The IMT runtime supports an abstract repository interface that allows permanent storage of objects and can be implemented in many ways. An interface to SQL relational databases is currently used to provide permanent storage on top of PostgreSQL, with full use of the database extensions (such as spatial operators and GIS support) that are currently lacking from open source native XML databases. The latter can be easily supported by implementing the abstract repository interface appropriately and loading the resulting code dynamically.

The IMT prototype is written in a modular fashion using policy-based design (Alexandrescu, 2001). This allows the runtime behavior to be customized in many ways. As an example, it is possible to create a IMT runtime that operates as a command-line application or as a threaded server by changing one line of code. Critical system

components such as the XML parsing engine, the permanent storage interface, or the support for multiple concurrent sessions can be customized in the same way. To interact with a IMT runtime that is running as a server, a generic client software is also available that can be customized to suit different modes of interaction. A specialized client program operates following the Common Gateway Interface (CGI) conventions and communicates results through HTML, giving full access to the functionalities of the IMT server through a web browser. This approach is being currently used to develop two projects, sponsored by the National Science Foundation:

- the Ecosystem services database (Villa et al, 2002; ESD, URL), a web-accessible database integrating data about the economic valuation of ecosystem services, the models that produce the values and allow their projections in the future, and the valuation methods themselves as models, allowing users to apply valuation methods to different data as the database grows.
- An integration of the existing African Mammals Databank (AMD, URL) and the pertaining entries in the IUCN Red List of threatened species, coupled with spatial data and predictive models, as a proof of concept in integrating data and predictive models in web-accessible databases.

The IMT can be easily extended with new classes of modules and domains that can support advanced functionalities. A major strength of the system is its ability to use and integrate the great amount of existing general-purpose open source libraries to achieve specific goals. Among the module classes developed or in development are:

- Wrapping classes that offer an interface to executable “legacy” applications, run as

“black boxes” by the module, while transparently handling the flow of data from and to the IMT dataspace.

- Optimizing modules (based on genetic algorithms and standard hill-climbing algorithms) that can run the modules they contain and optimize a set of goals stated by the user – e.g., obtain a parameterization that produces the maximum fit between the result modules and a specified set of data modules.
- Language interfaces to popular programming languages such as Scheme, Javascript, Perl and Python, to enable the definition of algorithms in the language of choice. One extension under development provides the option of compiling complex models into C and load the compiled object code dynamically for faster execution.
- An interface to the R statistical package (CRAN, [URL](#)) that allows complex statistical operations to be given a specific semantics and transparently become part of models and analytical pipelines.
- Modules that support dynamic difference equation models based on a stock and flow metaphor, with import filters that can translate models developed with popular modeling packages such as Stella (HPS, 1995; Costanza et al., 1998).
- Import plug-ins that allow to read and write data modules' states to and from popular data format such as ASCII tables, spreadsheets, GIS coverages;
- Import plug-ins that allow modules to define their observation context by reading formally specified metadata standards such as those endorsed by the Federal Geographic Data Committee (FGDC, [URL](#)) or EML (Partnership for Biological Informatics, [URL](#)).

## Perspectives

The IMA provides a semantic framework and a software infrastructure that unifies commonly separated entities and allows interoperability of independently developed representations of the natural world. The overall design principles have the ultimate purpose of enabling working with data and concepts using the same language that we use to think about phenomena and entities. Another fundamental goal is that any transformation can take place with known accuracy, and an estimate of the error implied by each transformation carried out is always available.

The proposed conceptual framework can have a direct effect on scientific creativity by removing many impediments to integration and testing of ideas, and producing faster feedback between conceiving ideas and testing results. IMA-enabled databases are ideal for intuitive and safe exploration of complex data and models.

I argue that through this and many other projects that are currently operating to extend the semantic awareness of natural systems research, we are nearing a switch in perspective and a shift in language that has the potential of getting rid of many artificial distinctions that currently clutter our vision of the concepts and prevent efficient and precise modeling of natural entities and phenomena. As an example, ecology has long been concerned with concepts like “modeling paradigms”, whose boundaries have rarely been crossed: “process-based models” and “individual-based models”, to mention only two of the most popular, have traditionally been separate areas of investigation (Villa, 1992). I argue that this can be as much the result of weak semantics as an actual

difference in approach. A representational framework that embodies a sufficiently expressive concept of scale has the potential of making many of these differences disappear. As an example, Figure 1 depicts a completely consistent model hierarchy, which contains both individual-based modules and traditionally process-based ones. We may be on the eve of developing new, simple representations that can encompass scales, representational paradigms and architecture-induced constraints and serve as a sound base for a semantic reorganization of the natural world which is computationally practical.

## **Acknowledgements**

The author acknowledges the generous support of the National Science Foundation, with grants 0243957 (Biological Database and Informatics), 0243962 (Biodiversity and Ecosystem Informatics) and 0225676 (ITR-SEEK). Many people have contributed to the development of the concepts expressed here. Among these, I must mention at least Serguei Krivov, Marta Ceroni, and the many participants to the SEEK project, most notably Bertram Ludaescher.

## References.

Alexandrescu, A. (2001). Modern C++ design: generic program and design patterns applied. New York, NY; Addison-Wesley.

Allen, T.F.H. and Starr, T.B. (1982). Hierarchy: Perspectives for Ecological Complexity. Chicago, IL; University of Chicago Press.

AMD (URL). The African Mammals Databank. Internet: <http://gorilla.bio.uniroma1.it/amd>.

Argonne National Laboratories (URL). Review of the Dynamic Information Architecture System (DIAS). Internet: <http://www.dis.anl.gov/DEEM/DIAS>.

Costanza, R., Duplisea, D., and Kautsky, U. (1998). Ecological modelling and economic systems with STELLA Introduction. Ecological Modelling 110: 1-4.

CRAN (URL). The Comprehensive R Archive Network. Internet: <http://www.ci.tuwien.ac.at/R>.

DMSO (URL). DoD High Level Architecture. Internet: <http://www.dmsomil/projects/hla>

ESD (URL). The Ecosystem Services Database. Internet: <http://esd.uvm.edu>.

FGDC (URL). Federal Geographic Data Committee. Internet: <http://www.fgdc.org>.

Fishwick, P. (1995). Simulation Model Design and Execution: Building Digital Worlds. Upper Saddle River, NJ; Prentice Hall.

Fritzson, P. and Engelson, V. (1998). Modelica - A unified object-oriented language for system modelling and simulation. In: Proceedings of European Conference on

Object-Oriented Programming (ECOOP98), Brussels, July 20--24, 1998. Internet:  
<http://citeseer.nj.nec.com/fritzson98modelica.html>

GBIF (URL). Global Biodiversity Information Facility. Internet: <http://www.gbif.org>.

Grundy, J.C. (2000). Multi-perspective specification, design and implementation of software components using aspects. *International Journal of Software Engineering and Knowledge Engineering*, 20(6).

HPS (1995). STELLA User's manual. High Performance Systems.

Ludaescher, B., Gupta, A., and Martone, M.E. (2001). Model-based mediation with Domain Maps. In: *Proceedings of 17<sup>th</sup> Intl. Conference on Data Engineering (ICDE)*, Heidelberg, Germany, 2001, IEEE Computer Society.

Minar, N., Burkhart, R., Langton, C. and Askenazi, M. (1996). *The Swarm Simulation System: a Toolkit for Building Multi-agent Simulations*. Santa Fe Institute Working Paper 96-06-042.

Mosterman, P. and Vangheluwe, H.L. (2000). Computer automated multi paradigm modeling in control system design. In: A.Varga, (Ed.) *IEEE International Symposium on Computer-Aided Control System Design*, 65-70. Anchorage, Alaska; IEEE Computer Society Press.

Partnership for Biological Informatics (URL): *Ecological Metadata Language (EML)*.  
Internet: <http://www.ecoinformatics.org/software/eml>.

Stevenson, R. D., W. A. Haber, and R. A. Morris (2003). Electronic field guides and user communities in the eco-informatics revolution. *Conservation Ecology* 7(1): 3.  
Internet: <http://www.consecol.org/vol7/iss1/art3>

TDWG (URL): International Working Group on Taxonomic Databases. Internet: <http://www.tdwg.org>

Vangheluwe, H.L., Kerckhoffs, E.J.H. and Vansteenkiste, G.C. (2001). Computer automated modelling of complex systems. In: Kerckhoffs, E.J.H. and Snorek, M. (Eds.): 15<sup>th</sup> European Simulation Multi-conference (ESM), 7-18. Prague, Czech Republic ; Society for Computer Simulation International (SCS).

Villa, F. (1992). New computer architectures as tools for ecological thought. *Trends in Ecology and Evolution*, 7: 179-183.

Villa, F., and Costanza, R. (2000). Design of multi-paradigm integrating modelling tools for ecological research. *Environmental Modelling and Software*, 15: 169-177.

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

Villa, F. (URL). Integrating Modelling Architecture home page. Internet: <http://www.integratedmodelling.org>.

Villa, F., Wilson, M.A., DeGroot, R., Farber, S., Costanza, R., Boumans, [R.M.J.](#) (2002). Design of an integrated knowledge base to support ecosystem services valuation. *Ecological Economics*, 41: 445-456.

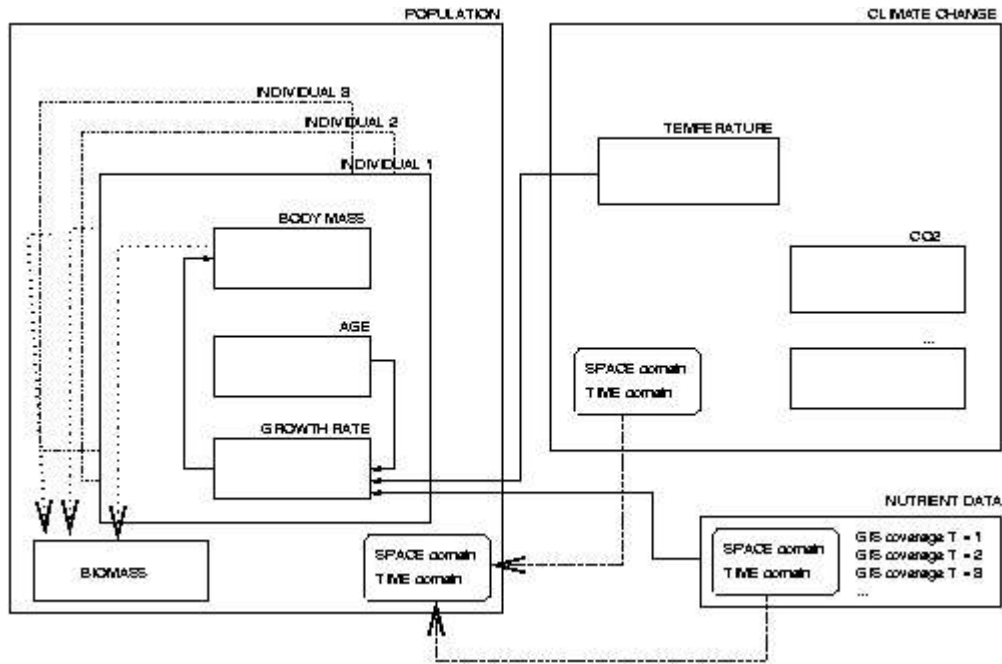


Figure 1. An hypothetical model hierarchy and its internal relationships. Boxes represent modules and containment relationships. The solid arrows mean dependency, the dotted arrows mean aggregation by containment, and the dash-and-dot arrows mean inheritance or compounding of a set of domains.