Agricultural Systems 103 (2010) 463–477

Contents lists available at ScienceDirect

Agricultural Systems

journal homepage: www.elsevier.com/locate/agsy

Ontology-based simulation in agricultural systems modeling

Howard Beck [a,*], Kelly Morgan [b], Yunchul Jung [a], Sabine Grunwald [b], Ho-young Kwon [b], Jin Wu [b]

[a] Agricultural and Biological Engineering Department, Institute of Food and Agricultural Sciences, University of Florida, Gainesville, FL 32611, United States
[b] Soil and Water Science Department, Institute of Food and Agricultural Sciences, University of Florida, Gainesville, FL 32611, United States

ARTICLE INFO

Article history:
Received 13 February 2009
Received in revised form 16 April 2010
Accepted 30 April 2010
Available online 1 July 2010

Keywords:
Ontology-based simulation
Ontology management system
Ontology
Simulation
Model
Citrus

ABSTRACT

A methodology and applications of ontology-based simulation are presented. An environment for building simulations based on the Lyra ontology management system is described which includes web-based visual design tools for constructing models and automatically generating simulation code. The ontology is used for representing all equations and all symbols appearing in these equations that are needed to describe a model. The example applications presented are models of soil, water, and nutrient management in citrus and sugarcane. Results thus far show that the ontology-based approach has advantages for representing the model structure, equations, and symbols, that complex models can be described in this format, and that efficient simulation code can be generated automatically from the ontology definition of the model. Potential applications, not yet fully explored, include ability to automatically connect models and data sources, using the ontology to organize model bases containing many models and model components, and using ontology reasoners to search for models, automatically discover model similarities and differences, and generate model instances from general principles.

Published by Elsevier Ltd.

1. Introduction

There is a need to better communicate all aspects of model structures and elements to the worldwide community of model builders. There is also a need for models to communicate and interact automatically on a machine-to-machine basis over the Internet. Modeling methodology can be improved by utilizing ontologies for all aspects of model building including the design, documentation, development and deployment of models. Application of ontologies to modeling and simulation is based on a new approach called ontology-based simulation. This paper explores several ways in which ontology-based simulation can be applied, as illustrated through specific applications in the area of soil, water and nutrient management.

An ontology can be defined as an "explicit specification of a conceptualization" (Gruber, 1993). An ontology is a formal representation of concepts and their relationship within a particular domain. In this case, the domain is an agricultural or natural resource system including hydrological, biological and physical transformations, and transport processes. The ontology in this domain includes concepts such as plant, soil profile, soil layer, water content, nitrogen concentration, and many others. Ontologies attempt to precisely define each concept (what water concentration is, how it is measured), and much of this is expressed through relationships among concepts (e.g. how is water concentration related to water content and soil volume). Ontologies are based on formal languages, meaning the concepts and relationships are expressed in a language that is well defined and not susceptible to ad hoc interpretation. A leading World Wide Web Consortium standard for such languages is OWL, the Web Ontology Language (OWL, 2005). Ontologies can also be thought of as machine-interpretable dictionaries since they provide formal definitions for domain concepts. Machine-interpretable means that the computer can make automated inferences about the relationships among concepts. Ontology reasoners have the potential to provide some additional functions including automatic classification and discovery. In ontology-based simulation, model building is considered to be a knowledge representation problem and not a software engineering problem. Software engineering is a formal technique for designing and building software systems. Traditionally, building a model involves writing computer code in a particular programming language. While advances in programming languages, including object-oriented programming and even the most recent Unified Modeling Language (UML) methodologies (Ivar et al., 1998) such as Model-Driven Architecture (MDA) (Raistrick et al., 2004), have been used to improve the process, most modeling is still viewed from the standpoint of how to best implement software to realize the model. In ontology-based simulation, the problem of modeling is raised to the level where the software implementation and associated software engineering concerns are irrelevant to the modeling process. Modeling becomes an abstract design problem in how best to represent knowledge about the system structure and

* Corresponding author. Address: Building 162, PO Box 110495, Gainesville, FL 32611-0495, United States. Tel.: +1 352 392 3797; fax: +1 352 392 4092.
E-mail address: hwb@ufl.edu (H. Beck).

0308-521X/$ - see front matter Published by Elsevier Ltd.
doi:10.1016/j.agsy.2010.04.004

behavior. Although ontologies continue to advance the tradition of object-oriented design, ontology languages are not programming languages. Ontology objects contain no variables, methods or other program code. They are purely declarative descriptions of concepts.

As with all object-oriented design methodologies, ontologies adopt a systems analysis approach through which complex systems are decomposed into smaller, interacting elements. In ontologies, dynamic behavior of physical and biological systems can be described not by program methods or constraint languages but through mathematical equations. Ontologies decompose systems to their smallest elements, resulting in equations and the symbols appearing in those equations. But simple objects recompose to form complex objects at all levels from the fine-grain symbols to the whole-systems view.

Ontology-based simulation addresses several problems associated with modeling and simulation. These include management of model libraries called model bases. System structure at a coarse level can be described by ontology objects as well. The problem of attaching models to data sources is addressed by building an ontological description of available data sources. Ontology reasoners can be applied to automatically classify, compare, and locate model elements to address problems related to model management. Solutions to these problems, some of which we have implemented and others which are proposed directions, are described in greater detail below.

Our ontology-based simulation environment is implemented within an ontology management system (OMS) that provides a platform and tools for creating and managing ontologies, including those used for simulation. We have constructed an OMS, which we call Lyra, for addressing a wide range of information management requirements in the area of agriculture and natural resources (Beck, 2008). Lyra includes support for modeling and simulation. The ontology management system is a database management system built entirely around an ontology language rather than traditional relational or object database languages.

We illustrate many of the concepts through a citrus water and nutrient management system (CWMS), and a similar system for sugarcane (OntoSim-Sugarcane) for modeling soil, water and nutrients with respect to soil physics and chemistry and demand for water and nutrients by the citrus tree. In the sections that follow, an overview of ontology-based simulations and the problems addressed is provided. Following that is a description of the Lyra OMS, and how CWMS and OntoSim-Sugarcane were implemented in this environment. We conclude with recommendations for future directions in the developing field of ontology-based simulation.

2. Ways in which ontologies can be applied to modeling agricultural and natural resource systems

2.1. What is an ontology?

An ontology is a formal specification of the concepts and relationships among these concepts within a particular domain. Concepts and relationships are defined using an ontology language. The language is formal in that it is well defined, but typically ontology languages are relatively simple in that they contain a limited number of language constructions. These constructs are designed to optimize a tradeoff between maximizing expressive power while supporting computationally efficient reasoning. We will introduce some basic ontology language constructions using OWL. All ontology languages include the notion of a concept. A concept can be generic, in which case it is represented as a *class*, or it can be a specific thing typically belonging to one or more classes, in which case it is represented by an *individual* (in OWL, an individual plays the role that is typically defined as an instance in other object-based languages). Individuals are the actual things in the world, and classes are categories that group together similar individuals and describe how they are similar. The set of all classes in an ontology comprise what is referred to as the "*T-BOX*", or terminology box, because the classes define the terms used in the domain. The individuals belong to an "*A-BOX*", or assertion box, which are statements about the actual things in the world, expressed using terms defined in the T-BOX. The word "object" is not part of the OWL language, but is often used to refer to individuals, or more loosely to both individuals and classes.

Classes and individuals are further defined through properties and other relationships. Generalization is described using superclass and subclass relationships with superclasses being more general forms of a class, and subclasses being more specific. This gives rise to the well-known generalization taxonomy common in most object-oriented languages. Individuals can have properties that describe the qualities of the individual (qualities are primitive properties such as strings and integers) and relationships to other individuals. A property has a domain (source) and range (target). The domain is the set of individuals for which the property can possibly be asserted, and the range is the set of possible values being asserted for the property. Classes do not have properties, rather they have property restrictions, which are constraints on what properties an individual must have to be a member of the class.

Typical reasoning facilities provided by ontologies include subsumption and classification. Subsumption is used to determine whether one class is a superclass of another. A subsumes B if B is a subclass of A, and logically implies that every instance of B is also an instance of A. Using subsumption tests, it is possible to automatically determine where in the taxonomy a new class should be placed (below the most specific classes that subsume the new class, and above the classes subsumed by the new class), a process known as classification. There are other possible reasoning facilities. Realization determines automatically whether individual C belongs to class A. Conceptual clustering can be used to cluster together similar individuals by identifying how individuals are similar or different. Classification can be used for query processing. A query is represented as a new class, the "query" class, and is classified to identify individuals belonging to the new class. However, much work remains in applying these techniques, and some potential applications are described below.

2.2. System structure

System structure can take many forms. One form is the logical system decomposition into sub-systems. A plant may have subsystems for processes, sub-structure, and temporal phases including photosynthesis and carbohydrate maintenance, water uptake and transpiration, phenological development stages, damage caused by diseases and pest, and others. There is also geometric structure. A plant is composed of leaves, stems, roots, and flowers. Soil topology can be decomposed into soil profiles and individual soil layers within a profile. The sub-systems comprising the system structure can all be described as concepts in an ontology. Relationships between sub-systems can be modeled as relationships between ontology objects. Geometric structure can be modeled using "part-of" relationship. For example, a soil layer is part of a soil profile. Categories of sub-systems, such as the familiar source, sink, storage, and flow elements of Forrester diagrams (Forrester, 1971), can be modeled as ontology classes.

The ability to decompose a system into smaller, simpler subelements is not unique to ontologies; it is an approach that has been used extensively in object design and by modular approaches

to modeling. However, ontologies enhance the approach by providing formal definitions and more precisely representing these subsystems. Ontologies enable concept sharing across modeling communities in spite of terminology differences. For example, the terminology can be made more precise, as in representing the difference between a soil layer and a soil horizon, and the terminology can change as needed by local convention. The terms soil layer and soil horizon may seem synonymous at first, but depending on context they can have different meanings. Soil horizon can refer to the vertical differences in soil characteristics which typically form discrete zones, whereas soil layer can be an abstract vertical dimension with a fixed upper and lower limit that can be different from the actual soil horizons. This subtle difference can be formally represented in the ontology because soil horizon not only has an upper and lower limit in a particular instance of soil, but also has associated physical and chemical properties that are not defined for an abstract soil layer based only on dimension. A soil horizon also has defined categorizations (A horizon, B horizon, etc.) whereas soil layer does not. These appear in the ontology as sets of property restrictions and associated classes.

2.3. Representing symbols and equations

All models of dynamic systems in agriculture and natural resources can be defined by a set of mathematical equations. By taking one additional step, using the ontology to represent the equations, we can describe model behavior within the ontology. Although the ontology cannot perform calculations necessary to implement a simulation of the model, the process of generating and running a simulation can be fully automated by using code generators once the model has been defined. There are many existing systems that enable modelers to design and build models at the mathematical level such as Mathematica (Wolfram, 2007) and Simile (Simulistics, 2007). Most of these include specialized languages used to define the mathematics. In our approach it is possible to express models using classic mathematical notation, and also gain the benefits of using an ontology to better define equation symbols.

Equations are composed of symbols, some of which are operators. Symbols can be expressed in many ways. One would be the mathematical form; for example, "t" can by a symbol representing time. But the actual symbol used is just a term, the concept is what counts. Thus an ontology object for "time" could be termed using "t", "time", or even "Zeit" to use a multilingual example. Time can be in different units (calendar date or Julian date, hours, minutes, etc.), and it can be discrete or continuous. Whatever the symbol, all associated knowledge about the symbol can be represented in the ontology object. Note that traditionally, associated knowledge must be manually written into documentation ("t represents time"), but this documentation is only human readable, not machine readable. Even if "t" appears as the name of a variable in a conventional computer program, the computer processes this information only as the name of a location in memory (e.g. add one to the value stored at location t), and has no information that t represents time.

Symbols that are operators (+, −, ×, /, =, and many others) can be represented by ontology objects having associations to the arguments needed by the operator. For example, divide requires a dividend and divisor. Equal requires two arguments for the two things being equated (a left side and a right side). An equation represented by a set of ontology objects takes the form of a tree as shown in Fig. 1.

Once equations are represented in this form, they can be part of the ontology model base, and the ontology can hold many equations needed for a particular model or for many different models. Different models can share the same equations.

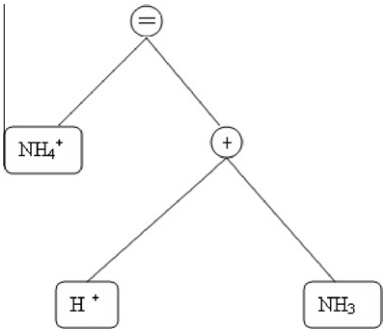


Fig. 1. An equation ($NH_4^+ = H^+ + NH_3$) represented in the ontology as a set of objects, one for each symbol. Relationships among operators results in a tree structure.

There are many possibilities for reasoning about equations represented in this form. One would be the comparison of equation structure. By comparing different trees (as in Fig. 1) for different equations, similarities and differences between equations used in different models could be determined. Also, equations in this form can be automatically converted to different formats. One of these is XML (Extensible Markup Language), thus equations can be converted to MathML (MathML, 2001) or OpenMath (OpenMath, 2000) for exchange with other systems. Another conversion is from equation to program code that can solve the equation. This technique is used to automatically build programs needed to execute the simulation and is one solution to the problem of how to perform the calculations. Although our experiments have been limited to Java, the same technique could generate code in C++, C# or FORTRAN.

2.4. Connecting to external databases

Models typically need to attach to external sources of data such as weather observations, data on soil characteristics, or information about production practices such as details of irrigation or fertilizer applications. Identifying existing sources of data, understanding their exact format, and adopting them to work within a particular model can be a tedious manual process. On the other hand, it may be possible in the future for models to search the Internet for suitable data sources and attach to them automatically. This could be done if databases are published as web services, and the web service registry provides sufficient information about a database to determine if it is suitable for a particular application. Ontologies can be used to represent database schemas and enhance searching within web service registries. For an existing relational database, each table becomes a class in the ontology, and each attribute within each table can be described by a property between the table class and another class that defines the range and units of the attribute.

Describing the database in this way would enable a model to automatically search for suitable databases. This would be done through a query that attempts to match up the class and instances for the database with symbols in the model (parameters and inputs). When a web service is located that provides a suitable database, the web service then makes the data available to the model in XML format through standard interfaces.

2.5. Integration with other information

Much additional information can be associated with a model. Documentation in the form of graphs and figures or additional text descriptions can be incorporated within appropriate objects in the model ontology. Data gathered through experiments for use in model estimation and validation can be included through a data-

base interface such as described in the previous section. Research publications associated with the model can be integrated within this framework as well. When simulations are used in training, the instructional design considerations, training scenarios, and assessment items can all be included (Badal et al., 2006; Beck, 2008).

2.6. Model base

A model base is a database of many models, model elements, equations, and symbols. There is a need for model bases as a way to organize the collection of models developed by many modelers over many projects. There is also a need to share and reuse models, sub-systems, and elements. In general, many similar but different models are developed by different modeling teams to address a similar problem (such as soil water balance). When these models share the same elements, it is desirable to identify and reuse those elements. It is also important to know where these models differ so that the appropriate model can be selected for use in a particular situation. During model development, it is convenient to be able to easily switch among different elements for modeling the same process in order to evaluate the behavior under each element. At the highest levels of abstraction, it is possible to show the most fundamental processes of physical systems and how they appear in different, lower level situations (e.g. a high-level abstraction such as capacity occurs in specific low-level situations such as water capacity for soil moisture, the concentration of nutrients in leaves of a plant, and in the capacitor of an electrical circuit).

The ontology facilitates construction and organization of model bases. The taxonomic organization allows for classification of different models and model elements at different levels of abstraction with the top of the taxonomy being most abstract and with more specific instances of models at the lower levels. Classification can facilitate organizing and locating model elements. Elements modeling the same subsystem or providing the same value for a symbol can be clustered within the same class. Fig. 2 shows a taxonomy of equations for the soil water and nutrient modeling domain. A particular model can select subsets of these equations as needed for a particular situation.

2.7. Ontology reasoning

Applications of ontology reasoning are based on comparison of object structure. This identifies how objects are alike or different. If one class is a special case of another, it can automatically be classified through a subsumption relationship. Similar objects can be grouped together to make new classes through conceptual clustering. These techniques can also be used for search and query processing by automatically identifying objects that satisfy a query class. Imprecise queries that locate objects that may be similar to but not identical with a target description are also possible. A technique for locating data sources was described in a previous section. Similar search techniques could be applied to locate existing model elements, equations or symbols satisfying some desired requirement.

2.8. Related work

Recently, ontologies have received much attention for implementing mathematical models and building simulation systems (Benjamin et al., 2006). Miller et al. (2004) noted that for modeling and simulation, an ontology provides standard terminology which increases the potential for application interoperability and reuse of simulation artifacts. Furthermore, semantics represented in an ontology can be used for discovery of simulation components, composition of simulation components, implementation assistance, verification, and automated testing. A web-accessible ontology for discrete-event modeling (DEMO) defines a taxonomy of models by describing structural characterization (state-oriented, event-oriented, activity-oriented, and process-oriented models) and a mechanism explaining how to run the model.

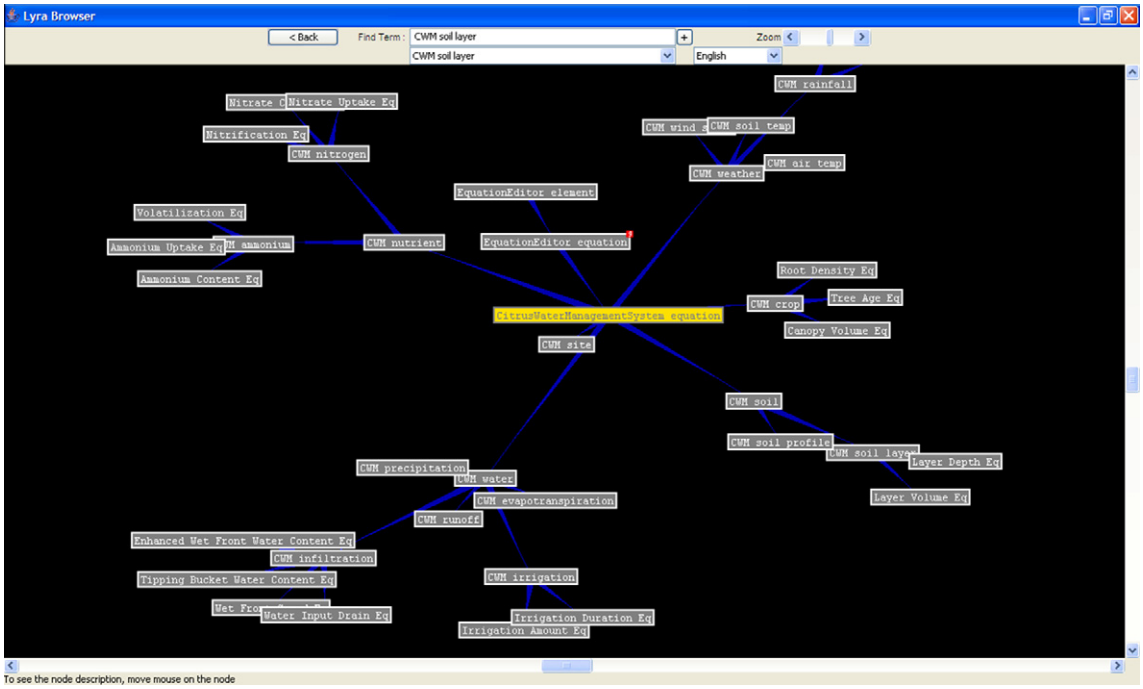


Fig. 2. Equation symbols for water and nitrogen balance visualized using the LyraBrowser. Symbols are represented as boxes, and an arrow means that the target symbol is required to calculate the source equation. Clusters associate symbols for rain, irrigation, evapotranspiration, hydraulic conductivity, and infiltration (left), nitrogen balance (right) and water balance (center).

Although Miller et al. focused on the creation of an ontology for general stochastic models such as Markov processes or Petri nets, Fishwick and Miller (2004) placed emphasis on capturing mostly object or instance-based knowledge. They presented a software framework, RUBE, which provides an integration method and multiple visual modes of display for developing dynamic models. 3D visualization was used by Park and Fishwick (2005) to animate the models. An ontology is used to define a schema of model types and models, and a sample air reconnaissance scene is represented with OWL.

Some research studies (Cuske et al., 2005; Jurisica et al., 2004; Raubel and Kuhn, 2004) use an ontology to describe entities in a simulation and the data and the rules governing the simulation. They argue that data used by a model is a key characteristic of semantics which an information system ontology should define. For example, ontology-based task simulation (Raubel and Kuhn, 2004) uses an ontology for evaluating the usability and utility of a task or data for the decision-making process. JOntoRisk (Cuske et al., 2005), which is an ontology-based simulation platform in the risk management domain, developed the meta risk ontology for validating or reviewing the meta structure.

SEAMLESS (Ittersum et al., 2008; Rizzoli et al., 2004, 2008) is a component-based framework for agricultural systems that is used to assess agricultural and environmental policies and technologies from the field-farm level to the regional level in the European Union. For SEAMLESS, an ontology is designed to relate different concepts from models, indicators and source data at different levels, and to structure domain knowledge and semantic meta-information about components for retrieving and linking knowledge in components. It also is used to check the linkage between components through input and output variables in the system. A Model Interface Ontology encapsulates knowledge of biophysical agricultural models. SEAMLESS does not attempt to represent models based on their mathematical equation form in the ontology.

A web-based simulation (Islam and Piasecki, 2008) uses an ontology of hydrodynamics. To solve the equations governing a two dimensional hydrodynamic model, an ontology was created to describe a numerical model and define a specific metadata set that describes hydrodynamic model data. Instances of a simulation ontology created during the simulation process are stored and retrieved by a relational database. The Modeling Support Tool, MoST (Scholten et al., 2007), a software framework for supporting the full modeling process, offers an ontological knowledge base (KB). The KB is a collection of knowledge on modeling for various domains of water management.

There is an effort to develop a set of crop models for various crops and integrating models with farm decision support system (Reddy and Anbumozhi, 2004). A modular approach to model development (Jones et al., 2001) contributes to categorizing and organizing models as software components, an executable unit of independent production (Donatelli et al., 2006a,b) in the agro-ecological domain. Caldwell and Fernandez (1998) mentioned problems on reuse of existing models in building a complex multi-level crop model and developed JanuSys, a framework supporting object-oriented conceptualization and hierarchical theory, for utilizing message-passing technique between model objects at different levels. Also, Moore et al. (2007) adopted a hierarchical concept to the Common Modeling Protocol (CMP), a generic and open framework for modular simulation modeling. It takes an explicitly hierarchical view of the biophysical system. However, they cannot fully address the difficulties of model management because they are developed for a specific programming environment and modeling framework, and they do not have the formality of ontology-based simulation techniques.

There have been several efforts to construct model bases, and recently ontologies are being applied to this purpose because of their strength in categorizing and organizing knowledge. Watershed modeling is considered as an aggregation system of unit hydrology and chemical processes, which includes precipitation, infiltration, evapotranspiration and erosion. Haan et al. (1982) presented a collection of generic processes and practical models which have been used to study the hydrologic cycle in watersheds. The MoST model ontology was developed following the structure of components in the system to manage models, and it made it possible to switch one model with other models in the same process level for seeking appropriate model composition resulting in an adaptable conclusion (Scholten et al., 2007). But, the complexity of the representation is not sufficient to describe processes in detail, and the large scale of the system makes it difficult to manage models. Although it enables model switching, it is limited to simple models.

Efforts are underway to examine the decision-making process over a farm region or water management area (Rizzoli et al., 2008; Scholten et al., 2007). This has resulted in a library of models that allows a user to build up a simulation system easily with unit process models. The library contains an ontology for storing the model knowledge which is gathered from references or experts. Models can be reused for building a larger system.

3. Example: soil water and nutrient management models

This section describes an environment which we have created for building ontology-based simulations. It is based on the Lyra ontology management system and includes web-based authoring tools for creating models. A full example of ontology-based simulation is provided using a model of soil water and nutrient management in citrus and sugarcane.

3.1. Lyra ontology management system

Lyra is an ontology management system which was created for the purpose of applying ontologies to a wide range of knowledge and information management problems in agriculture and natural resources (Beck, 2008). Lyra provides a solution to creating and managing ontology-based simulations and other applications where large number of objects must be created, manipulated, and efficiently stored, retrieved, and distributed. It includes the following capabilities.

3.1.1. Database management facilities

Lyra includes features commonly associated with database management facilities. Central to Lyra is an ontology language based on OWL. Lyra contains a set of language constructs for creating classes, individuals, property restrictions, properties, and data types. An efficient physical storage system optimized for ontology objects enables objects to be rapidly accessed and brought into main memory for processing. We are currently implementing ontology reasoners based on classification and subsumption, and we have implemented a SPARQL (W3C, 2007) query facility for filtering and searching. In order to publish ontology objects to make them available on the Internet, Lyra provides several web service and object request broker technologies. Lyra databases are wrapped inside Java Remote Method Invocation (RMI) servers that allow remote Java applications to attach to the server to send and retrieve objects. As a more standard language-independent solution, Lyra also supports web services that publish methods for sending and retrieving objects in XML format. Java servlets provide a simple URL-based technique for retrieving objects. Lyra supports full XML import and export so that the contents of the database can be shared. These capabilities allow application programs to access Lyra objects from anywhere on the Internet.

3.1.2. Authoring tools

Lyra supports a variety of authoring tools to enable modelers to directly create and manipulate objects. It includes some general purpose object editors (LyraBrowser and ObjectEditor), as well as application specific editors (RuleEditor, LanguageEditor, SimulationEditor and EquationEditor). The authoring tools have several common features. They are graphic-based, enabling modelers to manipulate objects visually. They are also web-based meaning that these tools are accessible through any web browser (that has the Java plug-in). The tools are cross-platform so that they run in many different browsers on different hardware platforms and operating systems. They all communicate to remote databases using web services, the Java RMI interface, or URL-based techniques. This results in a wiki-style collaborative development environment in which modelers at different locations around the world can work together to develop models.

LyraBrowser (Fig. 2) is a general purpose editor for visualizing and creating objects. An animated graphic interface allows authors to inspect objects and their relationships. Objects are displayed as nodes in a graph, and related objects are displayed using links. Authors can navigate the graph by clicking on nodes which then expand to show additional related objects. Editors for specifying object properties pop-up when authors click on object nodes.

ObjectEditor is similar to the LyraBrowser except that the displays are static. Sets of related objects are pre-arranged in a map that is created manually by the author. The database is segmented into modules of related objects.

RuleEditor is a domain-specific editor for creating expert systems. A rule editor allows authors to create IF–THEN rules using complex boolean expressions. A fact editor contains a list of facts and possible value. The RuleEditor automatically generates rule files that can be processed using the Jess inference engine (Friedman-Hill, 2007). Rules, facts, and associated symbols are stored in the Lyra OMS.

LanguageEditor is a domain-specific editor used for natural language processing. It enables creation of linguistic databases containing dialogs, phrase patterns (grammars), phrases, words, and morphemes.

SimulationEditor (Fig. 4) is an editor used to create system diagrams based on source, sink, storage, and flow components. This editor, along with the EquationEditor, is used to create ontology-based simulations.

EquationEditor (Fig. 3) is a template-based equation editor for creating equations and fully specifying associated symbols. It differs significantly from other equation editors in that the underlying representation of equations and symbols is defined by the ontology.

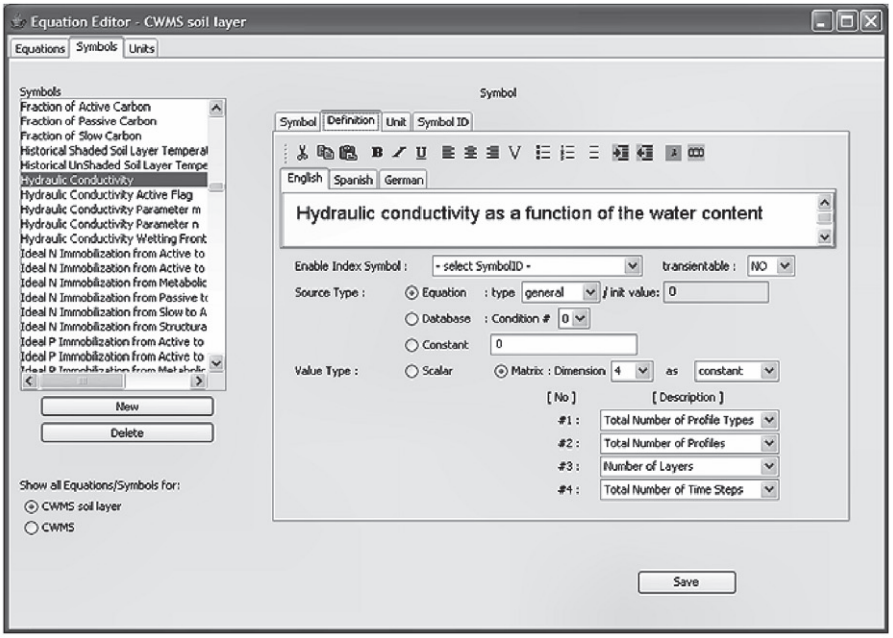
3.1.2.1. The EquationEditor. The EquationEditor (Fig. 3) is a tool for defining equations associated with a model and the symbols appearing in these equations. It provides facilities for creating, browsing, and inspecting all equations, symbols, and units appearing in the model. It uses an interface that resembles other equation editors such as Microsoft Office Equation Editor (Microsoft, 2003) and MathType (Design Science, 1996), but differs significantly because all the equations and symbols are represented internally using ontology objects. This provides a way to represent the meaning of equations and symbols that go beyond the informal techniques used with other equation editors. The EquationEditor is comprised of three sub-editors, symbol editor, mathematical expression editor, and unit editor.

The symbol editor is an editor for specifying individual symbols appearing in equations and includes a term expression of a symbol, a quantity of measurement, and a description of the linguistic and systemic properties of the symbol. A symbol is implemented as a class in the ontology which has a unique meaning within a specific domain. Often the same term (string of characters) is used over different domains and refers to different symbols and thus has different meanings. Since a symbol has a unique identifier and is associated with a specific concept in the ontology, use of the same term for different symbols is permitted, and the domain ontology can be used to resolve their ambiguous meaning. Properties associated with symbols (Fig. 3a) include the characters used to display the symbol, an English definition, units in which the symbol is expressed, a unique symbol identifier (Symbol ID), properties for identifying the symbol as scalar or matrix, and the symbol source (equation, database, or constant).

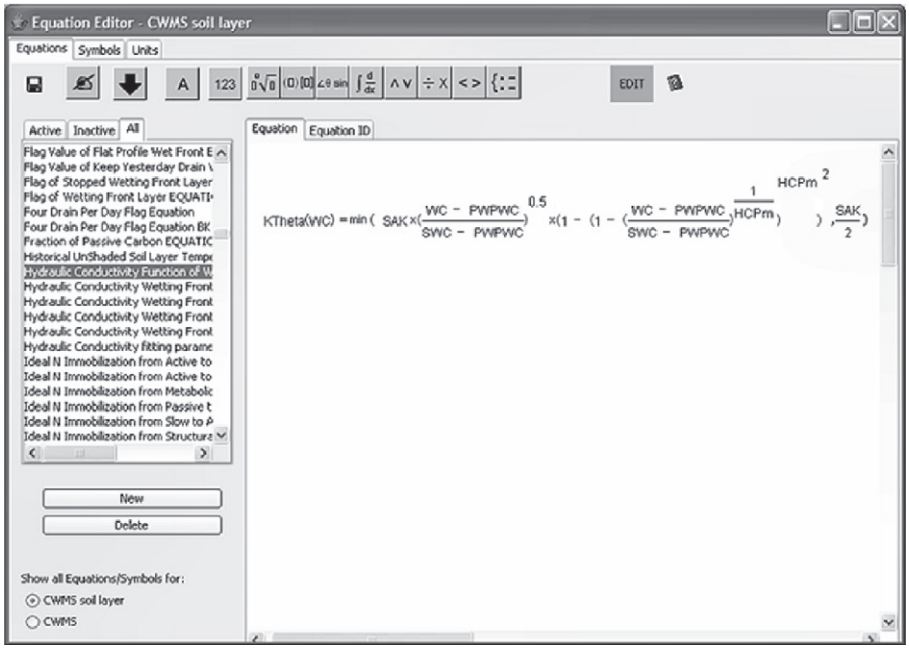
The numeric value of a symbol is obtained by one of three methods: from an equation, from a database, or from a constant which is directly assigned to the symbol. In the case where the symbol value is determined by an equation, there must be an equation in the ontology in which this symbol appears alone on the left side. To obtain the value from a database, some constraints are required in order to locate and query a database to obtain the value (e.g. a current time and a soil layer number for querying a soil temperature at a specific date), and these constraints can be specified as part of the symbol's properties. If the symbol value is a constant, the value of the constant is stored directly with the symbol. Symbols can also be arrays when a symbol is used in different discrete intervals in space and time. For example, soil–water content can be expressed in different soil layers which occur in different soil profiles, characterized by the depth from the soil surface, the soil profile number and time.

The mathematical expression editor is designed to graphically create an equation with mathematical operator templates and symbols. The editor provides more than 50 operator templates in eight operator groups used to compose an equation. An equation is an expression which internally has a hierarchical tree data structure composed of operators and symbols (Fig. 1). The equal operator is the root node of the tree. One constraint in the EquationEditor is that all equations have a single symbol on the left side. The value of the left side symbol is defined by the calculation of the right side expression. Thus the equation is assumed to be a function which has more symbols as arguments. The unit editor is used to create and maintain the unit for a symbol. A unit includes not only the generic collection of global standard units such as the metric unit system (SI), and the English unit system, but domain-specific units such as "cm^3 of soil". It is very important to carefully track the units associated with symbols since different models may use the same symbol but with different units. A unit is not represented by a simple string, but by a composition of symbols (like an equation). Though not yet implemented in the current system, this makes it possible to automatically convert one unit to another.

3.1.2.2. The SimulationEditor. Whereas the EquationEditor is used to design individual equations, the SimulationEditor is used to design and execute complete system models, and it uses the EquationEditor to define equations for each element in the system. The SimulationEditor is designed to represent models of dynamic systems using graphic elements such as source, sink, storage, and flow. It adopts concepts from the compartmental modeling technique (Peart and Curry, 1998) and Forrester notation (Forrester, 1971) which are widely used in agriculture and natural resource models. However, like the EquationEditor, these concepts are represented internally using the ontology. The SimulationEditor is used for specifying the overall model structure in the form of elements and incorporates the EquationEditor in order to build equations associated with each element. The SimulationEditor also contains facilities for automatically generating and running simulations and generating reports. The SimulationEditor provides a graphic user interface to create and maintain a simulation system which



(a) Symbol editor



(b) Mathematical expression editor showing the equation for hydraulic conductivity (KTheta) as a function of water content (WC), based on saturated hydraulic conductivity (SAK), saturated water content (SWC), permanent wilting point water content (PWPWC), and a hydraulic conductivity parameter (HCPm).

Fig. 3. Interfaces of the EquationEditor.

includes a structure design interface, a simulation control interface, and a simulation result reporting interface.

The structure editor is the main interface of the SimulationEditor and provides functionalities which enable a modeler to create and maintain a simulation project by designing the structure of a system and to interact with the EquationEditor and the simulation controller. Structural design of a simulation system is a procedure by which a modeler creates physical or environmental elements and relationships in the system by using graphic elements. The concept of graphic elements in the structure editor is based on the compartment elements of Forrester: source, sink, storage, and flow. For example, a 3-dimensional soil geometry may be described (Fig. 4) as a composition of soil cell (production unit), soil profile (horizontal), and soil layer module (vertical). These three elements may be defined as an instance of storage element, and relationships between these elements are represented by 'part of'

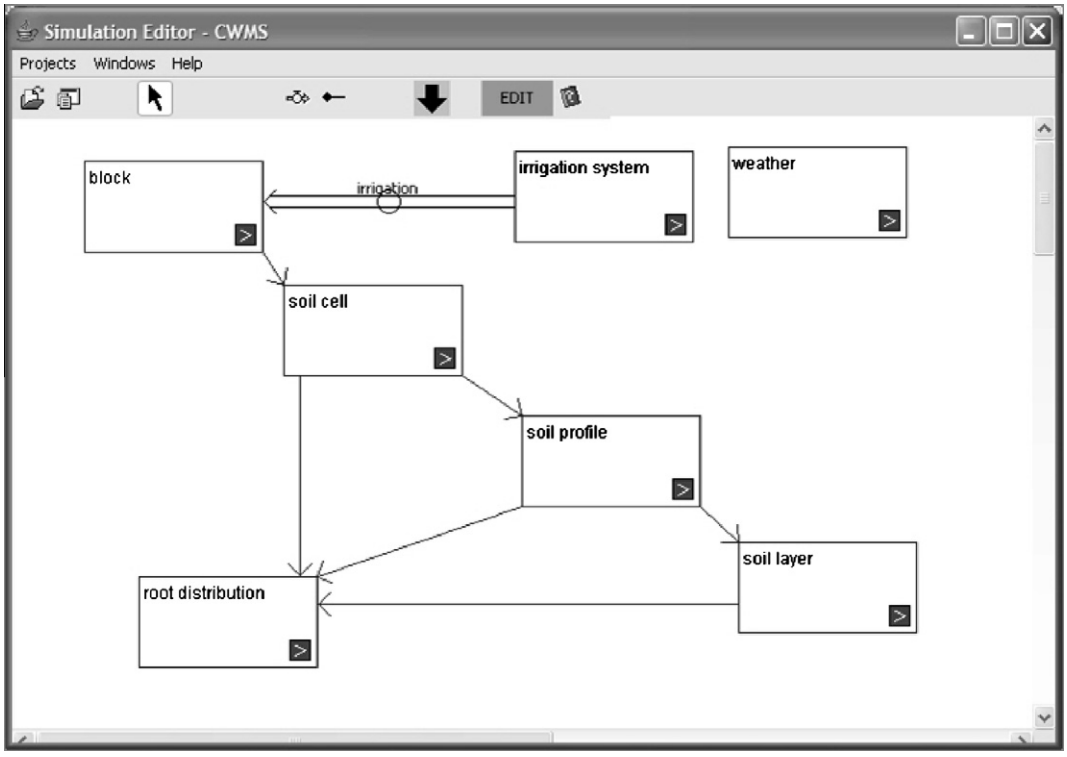


Fig. 4. CWMS system structure created using the SimulationEditor.

properties. Irrigation is realized with flow element representing the flow of water into the cell.

The simulation controller is a simulation engine used to generate and run a simulation based on the mathematical model under specified conditions with results presented in tabular or graphic displays. To generate a simulation, the simulation engine automatically converts ontology objects to program source code. It then compiles and runs the generated program to execute the simulation. Currently Java is the target language, although in theory code from other languages could also be generated. It is not necessary for the modeler to examine, work directly with or otherwise be concerned about the generated program source code. This process is completely internal to the operation of the software, and transparent to the modeler. The data object conversion and generation of program source code follows these steps:

- A class representing an element in the SimulationEditor forms a single class in Java. The class contains member variables and methods for all the symbols and equations in the element.
- Each symbol in the element is declared as a member variable named after the name of the symbol. If the symbol is an array, the member variable is declared as an array with the same dimensions as the symbol. A method is created that contains code for obtaining the value of the symbol. The name of this method is based on the name of the symbol. The value of the symbols is obtained from one of the following:
  - If a symbol is a constant, the return value of the method is a constant for the symbol's value.
  - If a symbol obtains its value from a database, the method returns a value obtained by querying a database for the value of the symbol, subject to constraints specified in the symbol's properties.
  - If a symbol obtains its value from an equation, the method contains code for solving the equation to obtain the value. Since the equation contains other symbols, these values of these symbols (on the right hand side of the equation), are obtained by calling methods for determining their values.

The generated source code can be used independently as part of a component library and inserted into other application. For example, the resulting simulation application can be integrated into a desktop application used by growers, it can be part of a larger decision support system such as DISC (Beck et al., 2006), or it can be part of a web-based simulation environment (users can run the simulation through a web page), or the simulation can deployed as a web service that is part of a distributed simulation environment.

Running a simulation involves compiling and executing the automatically generated source code. The simulation is controlled by recursively evaluating the value of a target symbol. Within the SimulationEditor, there is an interface to communicate with the model code library, which contains a method for calling the target symbol's method which results in execution of the simulation. The generated source code contains variables for storing all values of simulation symbols which are retrieved by the report generator to display model results.

The XML generator is a tool to generate an XML representation of the model. XML enables the model to be shared outside of the Lyra OMS environment. Two forms of markup language, MathML (MathML, 2001) and OpenMath (OpenMath, 2000), can be generated. MathML is an application of XML for describing mathematical notation and capturing its structure. It is designed for displaying mathematical formulas in web documents and other publications. OpenMath is a document markup language for mathematical formulae. Among other things, it can be used to complement MathML, which mainly focuses on the presentation of formulae, with information about meaning suitable for solving equations. To generate these XML formats from equations in the ontology, each operator template class which is declared in the EquationEditor has a method transforming operator and arguments to a string containing the

Fig. 5. The equation "A = B + C" is converted to MathML, OpenMath, and a Java method.

XML tag expression. An operator template can have other operator templates as arguments. The XML generator traverses the equation tree from the root operator template (which is always the "equal" operator) to each leaf symbol. An example of generating XML expression is shown in Fig. 5 for the simple equation "A = B + C" in both MathML and OpenMath format. A Java method for solving this equation is also shown.

3.2. Model implementation

Ontology-based simulation methodologies were applied to building a model describing water and nutrient balance processes for citrus production called the Citrus Water and Nutrient Management System (CWMS) (Morgan et al., 2007) and also to water and nutrient management in sugarcane (OntoSim-Sugarcane).To aid growers in management decision making, a computer-based decision support system was developed to facilitate more efficient use of water and nutrients by basing recommended application rates on site-specific characteristics and local weather data. We use these examples to show that relatively complex models, which have been validated and implemented with growers, can be constructed in such an environment, and that the process can benefit from the methodology we have presented.

3.2.1. CWMS

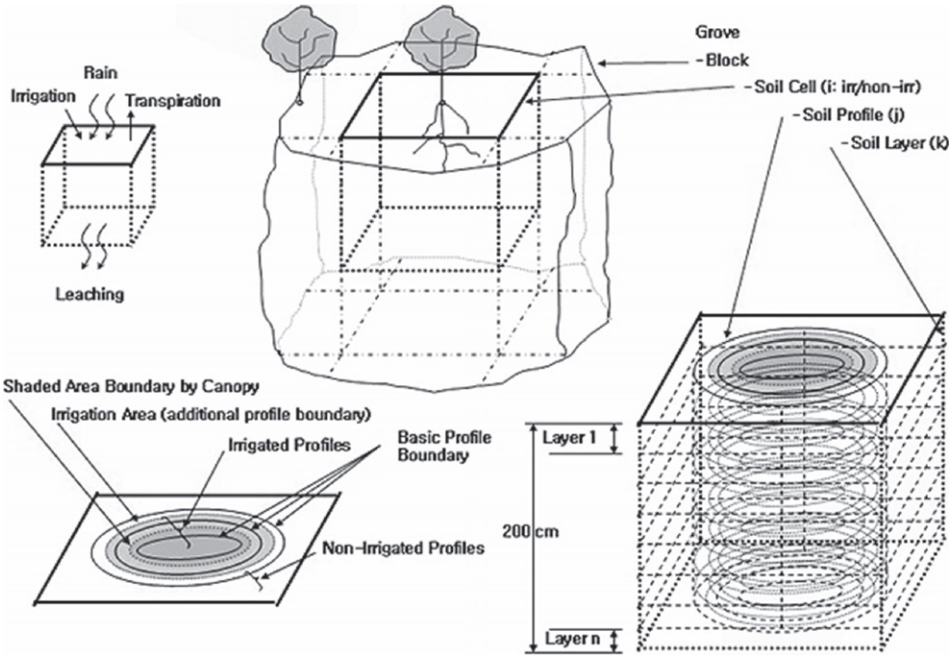
CWMS was constructed using the ontology-based simulation environment provided by Lyra. The entire model contains about 700 symbols and 500 equations. As illustrated in Fig. 6a (and shown formally in Fig. 4), seven concepts are defined for the model structure: block, soil cell, soil profile, soil layer, root distribution, irrigation system, and weather. To build a water balance model, a cell is defined as a cubical soil area surrounding one citrus tree, having a depth of 200 cm from the top of the soil. A cell is further divided into soil profiles within a cell, and soil layers within a profile. A commercial block of citrus consists of many cells since it has many trees, but in this model to simplify the simulation process the model is based on a single cell, and the single tree represented by that cell is characteristic of all the other trees in the block.

The width and length of the soil cell are described by the distances between trees in the row and trees between two adjacent rows. Each cell may have from 5 to 10 soil profiles which consist of 40 five cm thick soil layers. Each profile is designated as being one of four operational zones according to the irrigation and the dry-fertilized status: a non-irrigated & dry-fertilized area, an irrigated & dry-fertilized area, a non-irrigated & non-dry-fertilized area, and an irrigated and non-dry-fertilized area as shown in Fig. 6b. The physical and chemical characteristics of each soil layer are determined based on the particular soil series used in the simulation.

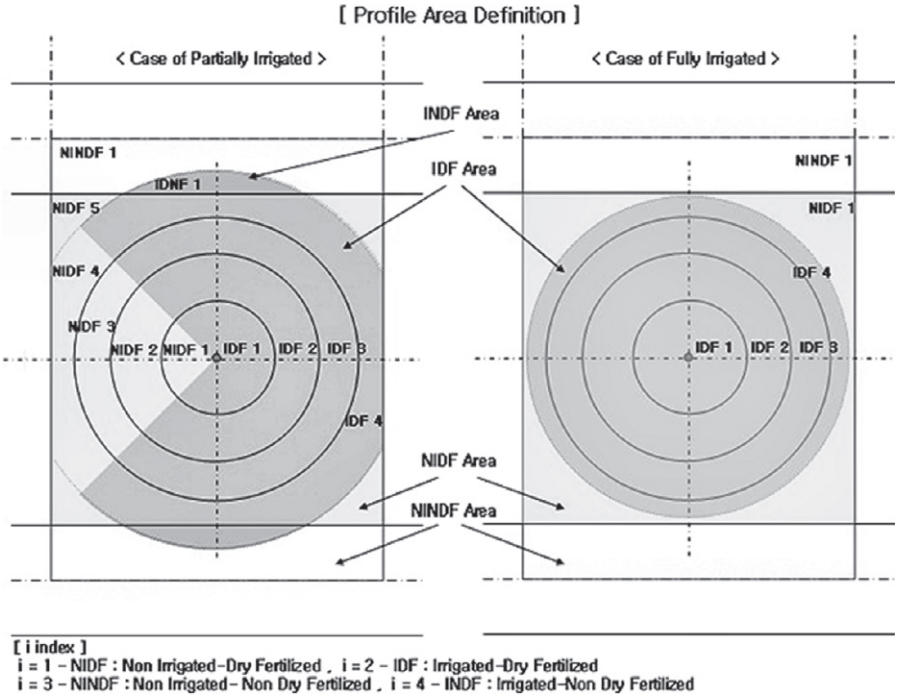
A soil layer is one vertical element of a soil profile, and the number of soil layers in a soil profile is determined by layer thickness and total depth of the soil profile, whose maximum depth is 200 cm. The spatial distribution of roots within the model is represented as a matrix of root densities by soil layer within each soil profile. The 3-D root distribution is based on root density distribution as a function of tree size (Morgan et al., 2006).

The CWMS model is an enhanced tipping bucket model. A pure tipping bucket model assumes that water moves through the entire simulation depth within one time step. CWMS does not make this assumption and simulates the wetting front movement among layers based on soil–water content within and below the wetting front, and the layer-specific hydraulic conductivity characteristics to determine the depth of wetting within each time step. Furthermore, since the exact times of irrigation or rainfall events are not inputs (this information is available only at a daily resolution), CWMS assumes that all irrigation, water with nitrogen (N) application and rainfall occurs at noon giving a maximum of 12 h to move through the soil on the first day after irrigation, water with N application or rainfall. The calculated wetting front speed and the time for which the wetting front travels a layer thickness are used to determine the wetting front depth at the end of the day. Water in excess of the soil's drained upper limit continues to drain to deeper soil layers the next day. These soil drainage processes, water infiltration, nutrient transformation and transport, and plant uptake are additional functions that are entered as interdependent equations along with values for soil characteristics (e.g. hydraulic conductivity, root density, and soil temperature) used by these main function equations. These soil characteristics change over time and must be recalculated for each layer of each soil profile with every time step.

Morgan et al. (2006) described how they derived the infiltration model used in CWMS from theoretical models. In summary, their infiltration model is based on Green-Ampt infiltration and unsaturated flow based on Richard's equation derived from Darcy's Law for irrigation and rainfall moving into soil from the surface and moving between soil layers. Furthermore, the model adopted Mein



(a) Soil cell (top center) contains one tree and is assumed to represent the entire grove, overall soil profile (bottom right) consists of multiple layers, and a further division of the overall soil profile (bottom left) includes several zones such as irrigated, non-irrigated, and shaded.



(b) Soil profile area designation for irritation and nutrient supply.

Fig. 6. CWMS geometry and soil structure.

and Larson's equation applying the Green-Ampt model for rainfall conditions by determining cumulative infiltration at the time of surface ponding (Fig. 7a).

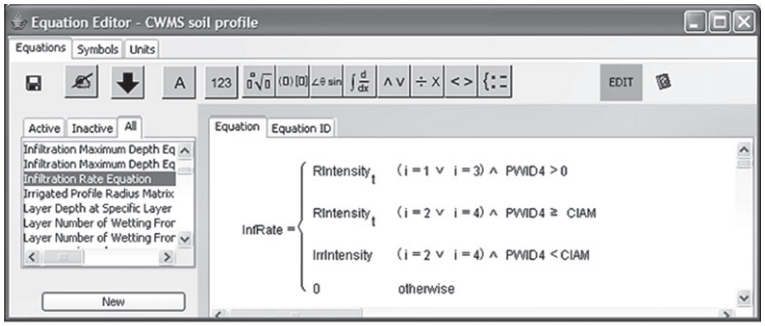
To make the model more applicable to the real world, additional condition terms describing whether the soil profile exists in the irrigated-area or non-irrigated-area were added to the model. With the above result, an equation for the infiltration rate of soil profile was formed in Fig. 7b. CIAM is the cell irrigation amount, and i is the profile (1, 2, 3 and 4), which includes a non-irrigated & dry-fertilized area, an irrigated & dry-fertilized area, an irrigated and non-dry-fertilized area, and a non-irrigated and non-dry-fertilized area. SAKtop is Ksat at the surface, and PWID4 is the soil profile water

$$f_p = R = K_s + K_s MS_{av} / F_p$$

$$\text{infiltrationRate} = \begin{cases} \text{rainIntensity } (t_{ponding} \text{ will be } \infty) & \text{if rainIntensity} < Ks \\ \begin{cases} \text{rainIntensity} & \text{if } t < t_{ponding} \\ \text{infiltrationCapacity} & \text{otherwise} \end{cases} & \text{if } Ks < \text{rainIntensity} \le \text{infiltrationCapacity} \\ \text{infiltrationCapacity} & \text{if } Ks < \text{infiltrationCapacity} \le \text{rainIntensity} \end{cases}$$

Where
Ks: Saturated Hydraulic Conductivity
t: time in hours
$t_{ponding}$: Ponding time when rainIntensity>infiltrationCapacity
infiltrationCapacity: decrease as t increase

(a) Theoretical form (Mein and Larson Equation) relating infiltration rate (f_p), rain intensity (R), saturated hydraulic conducitivity (K_s), water deficit (M), average suction at wetting front (S_{av}), and infiltration at surface ponding condition (F_p).



Where,

$$\text{RIntensity} = \min\left(0.8 \times SAKtop, \frac{1.75 \times R}{3}\right)$$

$$PWID4 = R \times (1 - \text{shadeAreaRatio} \times (1 - \text{shadedRainRatio}_t))$$

$$CIAM = IAM(IDur_t)$$

(b) Instantiated form for infiltration rate (InfRate) based on profile (i), rain intensity (RIntensity), irrigation intensity (IrrIntensity), water profile input (PWID4), cell daily irrigation amount (CIAM) and time (t).

Fig. 7. Example of reformulating the infiltration model from theoretical to instantiated form suitable for use in the context of CWMS.

input amount from rainfall effect. IAM is the equation calculating irrigation amount from irrigation duration, and IDur is the irrigation duration time.

3.2.2. OntoSim-sugarcane

OntoSim-Sugarcane (Kwon et al., 2010) is an application which models hydrology, nutrient cycling and crop growth on organic soils in southern Florida sugarcane production. Water table (WT) control systems and phosphorus (P) fertilization contribute to nutrient enrichment of the Everglades by releasing highly P enriched farm drainage water from the Everglades Agricultural Area (EAA).

The model includes processes for soil organic matter decay based on the mathematical framework of the CENTURY (Parton et al., 1988) as well as dynamics of inorganic P. Sugarcane growth and its nutrient uptake are modeled according to a modified version of the DSSAT-CANEGRO (Inman-Bamber, 1994). Hydrologic processes for simulating a perched WT and vertical and lateral drainage flux in the saturated zone were adapted from DRAINMOD (Skaggs, 1980) and other studies (Alexander, 1988; Chung et al., 1992; Reyes et al., 1993). These processes were represented as mathematical equations derived from the literature. In cases where processes were found in form of computer codes from various models (DSSAT-CANEGRO) we did reverse engineering from codes into equations.

These equations are implemented as database objects using SimulationEditor and EquationEditor, and then Java computer code is automatically generated using the SimulationEditor to run simulations. Finally, the successful compilation of the Java code was obtained after debugging errors that occurred while entering the equations.

3.2.3. Validation

Successful validation results are more a reflection of the quality of equations and parameters estimates used in the models, and do not prove any particular advantages of the ontology-based simulation environment. Nevertheless, a successful validation does show that the simulation environment accurately represented and rendered the models.

For CWMS validation, automated weather stations and soil moisture instrumentation were installed at six cooperator orchards to provide accurate rainfall, irrigation amount, ET, and soil moisture at 10, 20, 30 and 50 cm depths. Three stations were located in Sweet orange (*Citrus sinensis*, L.) orchards on deep sandy ridge soils in Highlands county, Florida and three stations were in orchards on poorly drained flatwood soils in Charlotte and Desoto counties, Florida. Data were collected over a 3 year period.

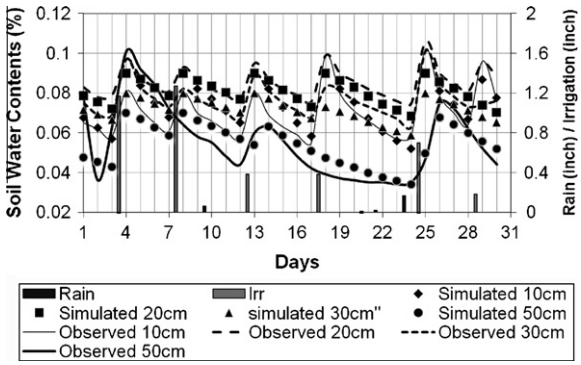


Fig. 8. Soil moisture content for a citrus orchard with continuous observed data (lines) and midnight simulated (model output) data (symbols).

Model estimated soil water content values were compared to soil water content measured by the capacitance sensors at midnight. Midnight values were used because the soil water content values estimated by the model assume irrigation or rainfall events at a length of time prior to midnight for most drainage to have occurred. Estimated model and measured soil content values were compared at 10, 20, and 30 cm depths in Fig. 8. Soil water content values in sandy soils are in a relatively narrow soil moisture range of 0.04–0.1 cm^3 cm^{-3}. This range represents the range of 66% depletion of available water to field capacity for most Florida sandy soils. Accuracy of the predicted soil water content was ±0.001, 0.001, 0.004, and 0.006 cm^3 water cm^{-3} soil at 10, 20, 30, and 50 cm depths, respectively (RMSE, Table 1). Efficiency index (IA, Eq. (1)) and coefficient of determination (R^2) decreased with increased soil depth, but remained above 0.88. Reduced accuracy of the model estimate with increased soil depth was probably due to upflux from the water table, reduced root uptake, and possible delays in soil water drainage.

$$\text{Efficiency index. IA} = 1 - \frac{\Sigma(\theta\text{sim} - \theta\text{obs})^2}{\Sigma[|\theta\text{sim} - \theta\text{meanSim}| + |\theta\text{obs} - \theta\text{meanObs}|]^2} \quad (1)$$

IA = efficiency index (0–1), θsim = model simulated soil water content, θobs = field soil water observation, θmeanSim = mean of simulation soil water contents, θmeanObs = mean of observed soil water content.

OntoSim-Sugarcane was validated for hydrologic processes with water quality monitoring data collected from two farms in the EAA, and good agreement between simulated and observed daily WT with the Nash–Sutcliffe efficiency coefficient (NSE; Nash and Sutcliffe, 1970) larger than 0.55 on the two EAA farms were observed. This indicated that OntoSim-Sugarcane is able to simulate daily fluctuations of WT within the farm units and estimate lateral drainage/sub-irrigation and deep seepage that significantly contribute to the water balance at farms in the EAA.

Table 1
Statistical comparison of model estimated and measured soil water content using 3 years of data for six citrus orchard locations.

	10 cm	20 cm	30 cm	50 cm
Soil depth				
RMSE[c]	0.0009	0.0015	0.004	0.006
IA[b]	0.9994	0.998	0.939	0.940
R^2[a]	0.9997	0.997	0.880	0.908

[a] Regression coefficient of determination (SAS Institute, Cary NC, USA).
[b] Efficiency index using equation '1'.
[c] Root mean square error for general linear model analysis of variance (SAS Institute, Cary NC, USA).

3.2.4. Application implementation

The CWMS model is used to implement a CWMS application program for use by growers that utilizes crop, soil and weather data. A CWMS application consists of the automatically generated simulation source code and a graphic user interface. Generated simulation code is plugged into the graphic user interface without modification.

A graphic user interface allows growers to interact with the system through three phases: the setup, the irrigation scheduling, and reporting. The setup phase, shown in Fig. 9a, is for configuring the cell and block information for a particular grove. The irrigation scheduling phase provides irrigation scheduling information to growers. By default, it is based on a 14-day simulation followed by a 3 day prediction period (the simulation period can be extended) to provide immediate term recommendations on irrigation rates. We have also run this simulation for full seasons (over 250 days per year, over multiple years) in order to do strategic planning on irrigation scheduling.

Detailed simulation results are provided in the form of daily, monthly, and yearly reports (Fig. 9b). The daily report contains each soil layer's root length, water content, nitrate and ammonium content, and soil coefficient. Data can be browsed by selecting a specific data and profile type. The monthly report shows data for a particular month including irrigation interval days and duration, evapotranspiration, crop coefficient, soil coefficient, water and nutrient leaching amount, irrigation depth, rain, and irrigation. The yearly report provides the monthly total values of irrigation, rain, water and nutrient leaching amount at 2 m depth and irrigation depth, and fertilizing amount. Water, nitrate and ammonium content contained in the daily report can be displayed as a chart.

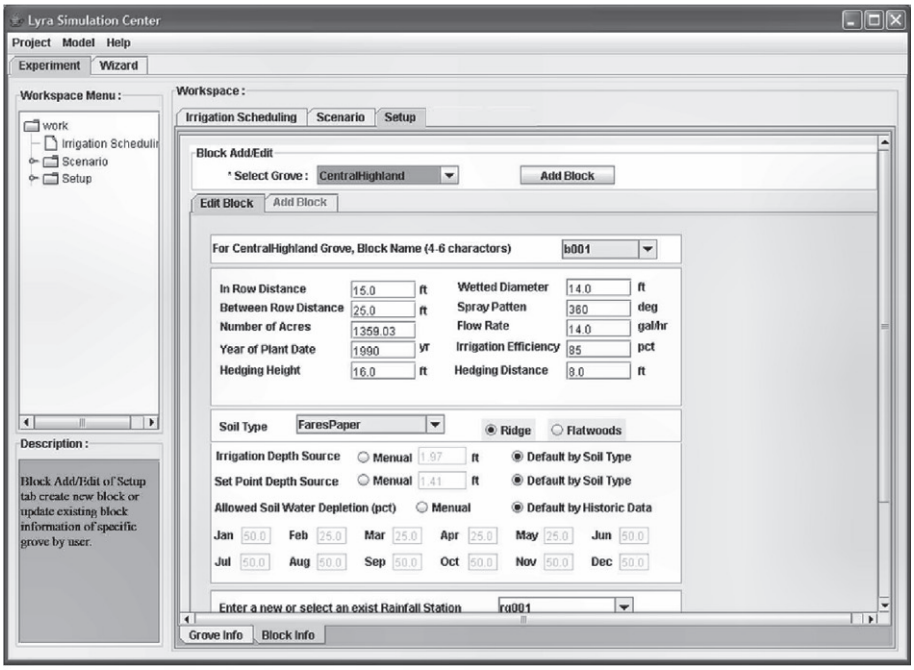
The CWMS Java code was generated automatically by the SimulationEditor and the EquationEditor, and can be used as a software component that is independent of the model building environment. The compiled Java code can be easily connected to other user interfaces or simulation system. For example, the Watershed Assessment Model (WAM) (Bottcher et al., 1998, SWET, 2006), which is a larger simulation of best management irrigation and nutrient management practices over entire watersheds, adopted the CWMS model as a sub-component of land use in citrus production. The CWMS model provides information about leached water and nutrient amounts to the host simulation system.

A key issue was how easily the CWMS model could be modified for integration with WAM. A water and nitrogen balance model needed by the WAM was required to use several new soil types, and modifications to the model were made to support these new parameters. Using the EquationEditor, new symbols for soil types and parameters were created and existing equations were modified by replacing old symbol and by adding new formula, and some new equations calculating required values by the WAM were added into the model. This was all accomplished with less than 10 h of work including creation of required file input/output protocol by WAM (that took about 80% of total work hours).

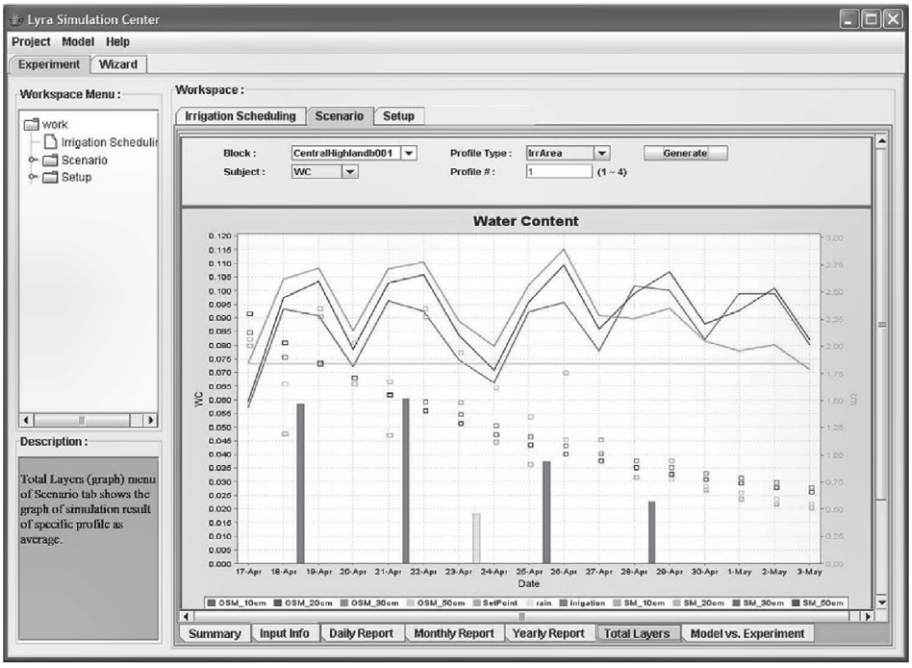
3.2.5. Model base

The approximately 700 symbols and 500 equations developed for the CWMS model are interrelated, and their relationship can be visualized as a graph diagram (Fig. 2), which displays connections of symbols and equations used to model the water and nitrogen balance process. Symbols are represented as boxes, and an arrow means that the target symbol is required to calculate the source equation.

In the diagram, the nitrogen balance process group is connected with the water balance process group by referencing several equa-



(a) System configuration



(b) Simulation results

Fig. 9. Features of the CWMS application.

tions for water balance. They can be switched with similar process groups independently. Particular processes, uptake, nitrification and volatilization, are clustered clearly from other equations in the equation group of nitrogen balance, and for water balance a similar pattern was found for processes of evapotranspiration, infiltration, and irrigation. This suggests possibilities for organizing and categorizing models and subprocesses.

3.2.6. Training and usability

While the simulation authoring tools have not yet been distributed beyond our development team (the other tools in Lyra are more widely distributed), some early assessment of usability can be reported based on experience of individual members of the team. The team consists of programmers developing the Lyra platform and authoring tools, modelers developing the specific soil water and nutrient models in citrus and sugarcane, and people doing field experiments for model calibration and validation. Only the two programmers were directly involved in writing code to implement the authoring tools. After getting the instructions for the use of the authoring tools for only an hour from the programmers, the modelers were able to work directly in the authoring tools without the need for knowing or working with the implementation code.

Symbols and equations can be easily created using EquationEditor once all processes required for the purposed simulations are obtained in the form of mathematical equations. A good example was the development of OntoSim-Sugarcane where 421 equations and 487 symbols related to soil–water–nutrient processes on organic soils in the Everglades Agricultural Area (EAA) of South Florida were created in the model within a few days.

In addition to this fast implementation of model processes using the authorizing tools, it was found that errors in the simulation were a direct result of errors in the equations, and not the source code. Thus, we maintain that it is not necessary to examine the automatically generated source code for any reason.

4. Conclusions

Ontology-based simulation has the potential to elevate modeling methodology in agriculture and natural resources to a level of abstraction where modeling becomes a knowledge representation processes and where reasoners can be applied to automatically classify, compare, and search for models and model elements. We have taken steps in this direction by building an environment for constructing models and representing equations and symbols in a formal ontology language, and we have demonstrated the utility of the approach by building moderately complex models of soil–water and nutrient management using this environment. This approach has the advantage of making models more explicit and better defining the meaning of symbols used in a model. Program code to run a simulation of our model is automatically generated from the ontology, and we can also export the model equations in XML formats including MathML and OpenMath.

To summarize the distinction between ontology-based simulation, object-oriented programming, and modular approaches to simulation design, it is necessary to first recognize that these approaches have much in common. They all are "object oriented" in the sense that they treat the world as being decomposed into smaller, interacting objects. The big distinction between ontology-based simulation and object-oriented programming is that the later is a software engineering approach, whereas the former is a knowledge representation approach that has nothing to do with, and indeed, explicitly avoids anything to do with programming languages and associated programming code. Whereas most modular simulation environments also deal with software components, ontology-based simulation makes such components explicit by providing a formal way to represent the semantics of the components as well as provide a representation of internal component parts down to a fine-grained level of resolution.

Much work lies ahead. We are currently exploring the use of the model base to integrate multiple related models by extending CWMS and OntoSim-Sugarcane to different soil conditions and different crops. We hope to show how the approach can be used to share model elements among models having similar sub-systems. The database connection described in Section 2.4 needs to be further developed. We are currently building web services for wrapping databases that can be integrated into models using this approach. There are many applications of ontology reasoners in searching for model components and comparing the structure of two different by similar models that need to be explored. Finally, sharing ontologies among modelers who collaborate on an international level has only just begun. Much discussion is needed to develop standards for sharing and coordinating ontology development at various levels within the modeling community.

Acknowledgement

This project was funded in part by a grant from the Florida Environmental Protection Agency, project number WM837.

References

Alexander, C., 1988. ADAPT – a model to simulate pesticide movement into drain tiles. MS Thesis. Department of Agriculture Engineering, Ohio State University, Columbus, OH.

Badal, R., Kim, S., Owens, J., Beck, H., 2006. An integrated database approach for managing educational resources in agricultural and biological engineering. IJEE 22 (6), 1210–1218.

Beck, H., 2008. Evolution of database designs for knowledge management in agriculture and natural resources. Journal of Information Technology in Agriculture 3 (1), 23.

Beck, H.W., Albrigo, L.G., Kim, S., 2006. DISC citrus planning and scheduling program. Proceedings of the Seventh International Symposium on Modelling in Fruit Research and Orchard Management. Acta Horticulture 707, 25–32.

Benjamin, P.C., Patki, M., Mayer, R.J., 2006. Using ontologies for simulation modeling. In: Proceedings of Winter Simulation Conference WSC 2006, Monterey, CA, December 3–6, pp. 1151–1159.

Bottcher, A.B., Hiscock, J.G., Pickering, N.B., Jacobson, B.M., 1998. WAM: Watershed Assessment Model for agricultural and urban landscapes. In: Presented at the 7th International Conference on Computers in Agriculture. Orlando, FL, October 26–30.

Caldwell, R.M., Fernandez, A.A.J., 1998. A generic model of hierarchy for systems analysis and simulation. Agricultural Systems 57 (2), 197–225.

Chung, S.O., Ward, A.D., Schalk, C.W., 1992. Evaluation of the hydrologic component of the ADAPT water table management model. Transactions of the ASAE (2), 571–579.

Cuske, C., Dickopp, T., Seedorf, S., 2005. JOntoRisk: an ontology-based platform for knowledge-based simulation modeling in financial risk management. In: European Simulation and Modeling Conference 2005, Riga, Latvia, June 1–4.

Design Science, 1996. MathType. <http://www.dessci.com/en/products/mathtype>.

Donatelli, M., Bellocchi, G., Carlini, L., 2006a. A software component for estimating solar radiation. Environmental Modelling and Software 21 (3), 411–416.

Donatelli, M., Bellocchi, G., Carlini, L., 2006b. Sharing knowledge via software components: models on reference evapotranspiration. European Journal of Agronomy 24 (2), 186–192.

Fishwick, P.A., Miller, J.A., 2004. Ontologies for modeling and simulation: issues and approaches. In: Proceeding of 2004 Winter Simulation Conference. Washington, DC, December 5–8, pp. 251–256.

Forrester, J.W., 1971. World Dynamics. Productivity Press, Cambridge, MA. p. 144.

Friedman-Hill, E., 2007. Jess, the Rule Engine for Java Platform. Sandia National Laboratories. <http://herzberg.ca.sandia.gov>.

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

Haan, C.T., Johnson, H.P., Brakensiek, D.L., 1982. Hydrologic modeling of small watersheds. ASAE Monograph 5, American Society of Agricultural Engineers, p. 553.

Inman-Bamber, N.G., 1994. Temperature and seasonal effects on canopy development and light interception of sugarcane. Field Crops Research 36, 41–51.

Islam, A.S., Piasecki, M., 2008. Ontology based web simulation system for hydrodynamic modeling. Simulation Modelling Practice and Theory 16, 754–767.

Ittersum, M.K.V., Ewert, F., Heckelei, T., Wery, J., Alkan Olsson, J., Andersen, E., Bezlepkina, I., Brouwer, F., Donatelli, M., Flichman, G., Olsson, L., Rizzoli, A.E., van der Wal, T., Wien, J.E., Wolf, J., 2008. Integrated assessment of agricultural systems – a component-based framework for the European Union (SEAMLESS). Agricultural Systems 96 (1–3), 150–165.

Ivar, J., Booch, G., Rumbaugh, J., 1998. The Unified Software Development Process. Addison-Wesley.

Jones, J.W., Keating, B.A., Porter, C.H., 2001. Approaches to modular model development. Agricultural Systems 70 (2–3), 421–443.

Jurisica, I., Mylopoulos, J., Yu, E., 2004. Ontologies for knowledge management: an information systems perspective. Knowledge and Information Systems 6, 380–401 (Springer-Verlag London Ltd.).

Kwon, H., Grunwald, S., Beck, H.W., Jung, Y., Daroub, S.H., Lang, T.A., Morgan, K.T., 2010. Ontology-based simulation of water flow in organic soils applied to Florida sugarcane. Agricultural Water Management 97, 112–122.

MathML, 2001. <http://www.w3.org/Math/>.

Microsoft, 2003. Microsoft Office Equation Editor. <http://office.microsoft.com/en-us/word/HP051902471033.aspx>.

Miller, J.A., Baramidze, G.T., Sheth, A.P., Fishwick, P.A., 2004. Investigating ontologies for simulation modeling. In: Proceedings of 37th Annual Simulation Symposium. Arlington, VA, April 18–22.

Moore, A.D., Holzworth, D.P., Herrmann, N.I., Huth, N.I., Robertson, M.J., 2007. The Common Modelling Protocol: a hierarchical framework for simulation of agricultural and environmental systems. Agricultural Systems 95 (1–3), 37–48.

Morgan, K.T., Beck, H.W., Scholberg, J.M.S., Grunwald, S., 2006. In-season irrigation and nitrogen decision support system for citrus production. In: Proceedings 4th World Congress on Computers in Agriculture and Natural Resources. Orlando, FL, July 24–16, pp. 640–654.

Morgan, K.T., Obreza, T.A., Scholberg, J.M.S., 2007. Characterizing orange tree root distribution in space and time. Journal of the American Society for Horticultural Science 132 (2), 262–269.

Nash, J.E., Sutcliffe, J.V., 1970. River flow forecasting through conceptual models part – A discussion of principles. Journal of Hydrology 10, 282–290.

OpenMath, 2000. <http://www.openmath.org>.
OWL, 2005. Web Ontology Language Guide. <http://www.w3.org/TR/owl-guide>.
Park, M., Fishwick, P.A., 2005. Integrating dynamic and geometry model components through ontology-based interface. Simulation 81 (12), 795–813.
Parton, W.J., Stewart, J.W.B., Cole, C.V., 1988. Dynamics of C, N, P, and S in grassland soils: a model. Biogeochemistry 5, 109–131.
Peart, R.M., Curry, R.B., 1998. Agricultural Systems Modeling and Simulation. Marcel Dekker, New York.
Raistrick, C., Francis, P., Wright, J., Carter, C., Wilkie, I., 2004. Model Driven Architecture with Executable UML. Cambridge University Press, New York.
Raubel, M., Kuhn, W., 2004. Ontology-based task simulation. Spatial Cognition and Computation 4, 15–37.
Reddy, V., Anbumozhi, V., 2004. Development and application of crop simulation models for sustainable natural resource management. In: Proceedings of International Agricultural Engineering Conference. IAEC, Beijing, PR China, October 11–14, pp. 10–35.
Reyes, M.R., Bengston, R.L., Fouss, J.L., Rogers, J.S., 1993. GLEAMS hydrology submodel modified for shallow water table conditions. Transactions of the ASAE 36, 1771–1778.
Rizzoli, A.E., Donatelli, M., Muetzelfeldt, R., Otjens, T., Sevensson, M.G.E., Evert, F.V., Villa, F., Bolte, J., 2004. SEAMFRAME, a proposal for an integrated modelling framework for agricultural systems. In: Proceedings of the 8th ESA Congress. pp. 331–332.
Rizzoli, A.E., Donatelli, M., Athanasiadis, I.N., Villa, F., Huber, D., 2008. Semantic links in integrated modeling frameworks. Mathematics and Computers in Simulation 78, 412–423.
Scholten, H., Kassahun, A., Refsgaard, J.C., Kargas, T., Gavardinas, C., Beulens, A.J.M., 2007. A methodology to support multidisciplinary model-based water management. Environmental Modeling & Software 22 (5), 743–759.
Simulistics, 2007. Simile <http://www.simulistics.com/products/simile.php>.
Skaggs, R.W., 1980. DRAINMOD Reference Report: Methods for Design and Evaluation of Drainage Water Management Systems for Soils with High Water Tables. North Carolina State Univ. Press, Raleigh, NC.
SWET, 2006. Watershed Assessment Model Documentation and User's Manual. Soil and Water Engineering Technology, Gainesville, Fla.
W3C, 2007. SPARQL. <http://www.w3.org/TR/rdf-sparql-query>.
Wolfram, 2007. Mathematica. <http://www.wolfram.com/products/mathematica>.