A QUALITATIVE SIMULATION FRAMEWORK IN SMALLTALK BASED ON FUZZY ARITHMETIC

Richard L. Olson, USDA-Agricultural Research Service, PO Box 5367, Mississippi State, MS 39762, USA
Daniel L. Schmoldt, USDA-Forest Service, Madison, Wisconsin, USA
David L. Peterson, Department of Forest Science, University of Washington, Seattle, Washington, USA

SUMMARY

For many systems, it is not practical to collect and correlate empirical data necessary to formulate a mathematical model. However, it is often sufficient to predict qualitative dynamics effects (as opposed to system quantities), especially for research purposes.

In this effort, an object-oriented application framework (AF) was developed for the qualitative modeling of natural-resources systems. An application framework is a collection of reusable object classes that provide basic functionalist y for a class of applications. The user creates subclasses that inherit this functionalist y, and that are specific to his or her application. Smalltalk, an object-oriented programming (OOP) language, was chosen for the AF.

The modeling methodology is based on Schmoldt (1991). Parameters represent variables of interest in the model system. The magnitude of a parameter is represented by its quantity, a fuzzy number, and the effects of changes in parameters upon one another are simulated over time by fuzzy arithmetic. The values of the changes in parameters and their quantities are then translated into linguistic values.

The AF includes classes providing all the functionality for constructing application-specific fuzzy simulation, including FuzzyNumber, Parameter, FuzzySimulation and FuzzyTimer. Also included are data-structures such as KeyValueList and AssociationList, and modifications to Smalltalk to smoothly integrate fuzzy arithmetic for non-fuzzy values. The framework runs under Microsoft Windows 3.1, and includes a graphical user interface. In this paper, we present the AF along with an example simulation of plant physiology.

KEYWORDS: qualitative simulation fuzzy

INTRODUCTION

Mathematical modeling has been a staple technique in the study of natural resources for some time. Such models can be broadly categorized as empirical or mechanistic

(Olson *et al.,* 1985). In empirical modeling, a mathematical equation is fit to a data set (or sets) in order to describe a phenomenon of interest. Such models are data-intensive, because they have little validity beyond the data-set from which they were derived. Therefore, to be applicable to new situations, they must be re-parameterized with a completely new data-set (Dale *et al.,* 1985) -- they are often described as being "brittle" (Holland, 1986).

Mechanistic models are those that are formulated based on biological relationships, usually at one level below that of the phenomenon of interest. Like empirical models, they are data-intensive, because their parameter-sets must be derived from experimental data. Because they are based on biological mechanism, however, they are nominally less brittle (i.e., more robust) than empirical formulations. Isebrands *et al. (1990)* reviews mechanistic tree growth models.

Although the methodologies differ, the level of data necessary for the use and construction of both empirical and mechanistic mathematical models is high. Many times, especially in the case of long-lived species such as forest trees. the data is simply not available. If information for a species is available, it is often only for a particular growth stage (i.e., immature). Any mathematical model of the entire life history must therefore be extrapolated from incomplete information. Finally, in complex simulation models small errors in parameter estimation can be magnified into major qualitative and quantitative errors at the level of interest.

The techniques used in any modeling effort should be appropriate to the desired end-uses of the model. If predictions of system quantities are desired, as in fire or growth-and-yield modeling, mathematical formulations are unavoidable. However, often a qualitative description of the changes in system quantities is all that is necessary. In these cases, the *patterns* of the system responses are of interest more than the exact *values* of those responses, and a qualitative model is more suitable than a quantitative formulation.

Schmoldt (1991) describes such a qualitative modeling technique based on fuzzy logic. In this paper, we review that technique, and then describe an object-oriented implementation of it in the Smalltalk programming language.

THE QUALITATIVE SIMULATION TECHNIQUE

Models using the Schmoldt (199 1) technique are called "influence models" because they can be represented as a diagram of the influences between model parameters. Figure 1 shows a fragment of the tree-physiology model from Schmoldt (1991). Arrows represent the influences of each parameter in the model, which are represented by boxes. Thus, the parameters *stomatal-aperture, chlorophyll-deficiency, total-needle biomass,* and *translocation* all influence the *photosynthesis,* which, in turn, influences *carbohydrate-accumulation.* The latter parameter influences both *assimilation* and *respiration.*

Each parameter in the model can be represented by a current value, or "quantity", and a current rate of "change". In a qualitative model, these take on linguistic attributes, such as "low" or "high" for quantity and "slightly increasing" or "moderately-decreasing" for change. In the model, each parameter has a minimal set of information stored with it (Figure 2): it's present quantity, rate of change, and a list of it's dependencies *(i.e.*, the other parameters in the model that affect it), accompanied by the direction of influence (+ or -).

Within the model, the linguistic values for "quantity" and "change" are represented as *fuzzy numbers.* These numbers, and the fuzzy calculus used to manipulate them, are derived from Zadeh's theories of possibility and fuzzy sets (Zadeh, 1965). Although we won't go into fuzzy-set theory in any detail [see Schmoldt (1991) for a description of ideas relevant to the model], we will discuss the concepts necessary to understand the modeling framework described in this paper.

Fuzzy Sets

In a fuzzy set, measurements in some domain are mapped to a set of linguistic variables, such as "large" or "small". For each fuzzy subset, a *membership function, μ,* assigns a probability y that each measurement belongs to it. For example, μ_{large} contains the probabilities that each measurement in the domain belongs to the fuzzy subset "large." In an example from Schmoldt (1991), Table 1 contains two membership functions, μ_{large} and μ_{small} for tree diameter measurements. From this table, it can be seen that the probability of a tree measuring 10 cm in diameter being in the subset "large" is only 0.2, whereas the probability that it is in the subset "small" is 0.8.

diameter	μ_{large}	μ_{small}
5 cm	0.0	1.0
10 cm	0.2	0.8
15 cm	0.4	0.5
25 cm	0.7	0.1
35 cm	0.9	0.0
45 cm	1.0	0.0

Table 1. Two membership functions (μ_{large} and μ_{small}) relating to diameter classes. After Schmoldt (1991). See text for details.

The fuzzy numbers "large" and "small" may be defined, with regards to diameter, from Table 1 as {0.0/5, 0.2/10, 0.4/15, 0.7/25, 0.9/35, 1.0/45} and {1.0/5, 0.8/10, 0.5/15, O. 1/25, 0.0/35, 0.0/45}, respectively. Note, that to be comparable, the numbers must have the same *basis,* in this case the set {5, 10, 15, 25,35, 45}. More generally, we can write any fuzzy number F as:

$$F = \{f(n)/n \mid i \leq n \leq j\} \qquad (1)$$

where f(n) is a membership value at basis value n within the basis bounded by i and j.

Table 2 lists the linguistic variables for the attributes “quantity” and “change”. The bases for these sets and their membership functions are beyond the scope of this article; they are defined in Schmoldt (1991).

Set	Value		
Quantity	Zero	Low	Moderate
	Moderately-high	High	Very-high
Change	Strongly-decreasing	Decreasing	Moderately-decreasing
	Slightly-decreasing	Steady	Slightly-increasing
	Moderately-increasing	Increasing	Strongly-increasing

Table 2. The linguistic values for “change” and “quantity”. See text for details.

In the simulation, when a parameter changes, it is propagated through the influence network via fuzzy calculus. Again, the description of this calculus is beyond the scope of this paper, but ‘is defined fully in Schmoldt (1991).

We next describe the object-oriented modeling framework that implements the model described above.

FUZZY-AF

The fuzzy simulation application framework (Fuzzy-AF) is an object-oriented modeling system that implements the qualitative simulation method of Schmoldt (1991). Object-oriented programming (OOP) is a powerful design paradigm that facilitates code reuse and modularity. We will not go into the details of OOP in this paper; for a good introduction to the topic as it applies to natural resources management programs, see Olson and Wagner (1992).

The *application framework* (AF) concept takes full advantage of the reusable and modular nature of OOP code. An AF is a collection of *abstract superclasses* (i.e., classes that will not be instantiated) implementing the basic functionality of a kind of application. An AF is used by creating subclasses that are specific to a user’s application. Application frameworks are common in today’s object-oriented programming systems; a well-known example is the Microsoft Foundation Classes (Microsoft Corp., Redmond, Washington, USA), a collection of C++ classes for the implementation of graphical user interfaces on personal computers. For more on the AF concept, and another example of it's use in biological simulation, see Olson *et al.* (1996).

Fuzzy-AF is implemented in Smalltalk, a pure object-oriented programming language (Goldberg and Robson, 1989) in the Visual Smalltalk dialect (ParcPlace-Digitalk Corp., Sunnyvale, California, USA). The framework runs on IBM-compatible computers of 80486 class or better.

The framework consists of four major classes, those that implement the modeling paradigm and the fuzzy mechanics, and five support classes. The latter implement special data-structures and simulation book-keeping. The major classes are: **Parameter, FuzzyNumber, FuzzySimulation,** and **FuzzyTimer.** The support classes include **Simulation, SimulationObject, AssociationList, IDAssociationList,** and **KeyValueList.** In addition, there are also overridden methods for the Smalltalk classes **Float** and **Integer** that handle mixed arithmetic for fuzzy numbers. Table 3 lists the classes in Fuzzy-AF.

Major AF Classes	Support Classes
Parameter	SimulationObject
FuzzyNumber	Simulation
Fuzzy Simulation	AssociationList
FuzzyTimer	IDAssociationList
	KeyValueList

Table 3. A list of major and support application framework Classes. See text for details.

Figure 3 shows the inheritance hierarchy for the major classes of Fuzzy-AF. Inheritance is a powerful feature of object-oriented programming -- it is what makes the application framework concept possible. Classes lower in the hierarchy inherit functions (methods) and data-structures (but *not* data values) from those higher. The arrows show the direction of inheritance; thus, Class **FuzzySimulation** inherits from Class **Simulation,** and so on. To illustrate the implementation of fuzzy simulation in Fuzzy-AF, we next describe selected classes in more detail, examining selected data structures and functions of each.

Class Parameter

Figure 4 shows a partial definition of Class **Parameter.** Data structures include *quantity* and change, fuzzy numbers representing the quantity and change values, respectively; *quantityLinguistic* and *changeLinguistic,* the linguistic equivalents of quantity and change values (eg., "low" and "moderately-increasing"); *influences* and dependencies: lists of pointers to instantiated parameters that the object influences and that influence the parameter, respectively; *influenceClasses* and *dependencyClasses,* lists of the types of influences and dependencies, respectively; and the *timeScale* on which the parameter operates.

Functions of Class **Parameter** include *update,* which updates the parameter if needed; *updateQuantity* and *updateChange,* which update the fuzzy numbers representing quantity and change, respective y; and *translate:with,* which translates fuzzy numbers into their linguistic equivalents.

Class FuzzyNumber

Figure 5 contains a partial definition of the class **FuzzyNumber.** There is only one data structure, *value,* a list of basis numbers and their associated values for the particular fuzzy number represented by the instance of the class.

There are six arithmetic functions supported by class **FuzzyNumber:** +, -, *, /, >, and *exp.* Each arithmetic function overrides the built-in Smalltalk binary function it replaces; for example, the function "+" first tests to see whether its argument is another fuzzy number. If so, it uses the addition rule for two fuzzy numbers outlined in Schmoldt (1991); if not, it uses the mixed number rule in that paper. This illustrates a major advantage of the object-oriented paradigm: functions that have identical names, but different implementations, can be defined for multiple classes of object. This feature, called *polymorphism,* enables the user to use the same function-names to access similar features in different object-classes without regard to how the are implemented.

The other functions in Fig. 5 are for the manipulation and modification of fuzzy numbers: *makeConvex* renders the number as a convex set; *interpolateBetween:and:* finds the membership values on a straight line between two base/value pairs in the fuzzy number; and *distance To:* finds the Euclidian distance between itself and another fuzzy number. All of these functions are necessary in the arithmetic manipulation of fuzzy numbers, and, the propagation of changes through the model influence net.

Class Fuzzy Simulation

Class **FuzzySimulation** (Fig. 6) stores all the information necessary for the running of a qualitative fuzzy simulation. Data structures include *partClasses* and *parts* (inherited from its superclass Simulation), which are a lists of the parameter classes in the model and references to instances of those classes, respectively; *changeDefinitions* and *quantityDefinitions,* the linguistic definitions for change and quantity values, respectively; *changeBasis* and *quantityBasis,* the bases for change and quantity values, respectively; and *timer,* a pointer to an instance of Class **Fuzzy Timer.** Instances of the latter class keep track of system time.

Functions of Class **FuzzySimulation** include *constructNet,* which builds the network using *partClasses* , the functions *constructDependencies* and *constructInfluences,* and the lists of dependency and influence classes found in each parameter; and *initialize,* which initializes the entire simulation.

We next examine some user issues, and take a brief look at how the simulation is accomplished.

USING FUZZY-AF

To develop a simulation the user first creates subclasses of Class **Parameter** that are specific to her/his application. Figure 7 shows the Fuzzy-AF inheritance structure, with subclasses specific for the tree physiology model (Schmoldt, 1991) emphasized. The classes thus created must have information relevant to the network diagram stored in the data-structures *influenceClasses* and *dependency Classes.* Thus, they "know" at run-time what their related parameters are.

The user must also create an application-specific subclass of Class **FuzzySimulation.** This class contains simulation-specific information about the bases of the fuzzy numbers used in the simulation, and the linguistic definitions for change and quantity.

To run the simulation, a new instance of the **FuzzySimulation** subclass is created using a Smalltalk message of the form

a := Fuzzy SimulationSubclass new, (2)

which creates a new instance of **FuzzySimulationSublcass** and assigns it to the variable "a". Upon its creation, the instance of **FuzzySimulationSublcass** does the following:

1. sets the change, quantity bases
2. sets the change, quantity definitions
3. constructs its parts list (i.e, initializes all parameters)
4. constructs the influence network.

To run the simulation for n time units, a Smalltalk command of the following form is issued:

a runFor: n (3)

For each time interval, the **FuzzySimulationSublcass** instance sends the message *update With:* t(where t is the system time) to each parameter. Each parameter can be updated at different time intervals, and so must decide whether or not to update itself. If so, it does the following:

1. updates the change values
2. updates quantity values
3. translates change and quantity fuzzy numbers into linguistic values

CONCLUSIONS

Due to a lack of required data and numeric inconsistencies, numeric simulations of biotic systems are often infeasible. When dynamic response and pattern prediction are all that are necessary for the purpose of the model, a qualitative simulation is sometimes sufficient. An Application Framework, Fuzzy-AF, for qualitative simulation using fuzzy arithmetic was developed based on the methodology of Schmoldt (1991). The framework is implemented in the Smalltalk programming language, and runs on 80486-class (or better) IBM compatible microcomputers. The framework is usable by others via the creation of user-specific subclasses of the application framework classes. Fuzzy-AF shows promise as a general-purpose tool for qualitative simulation in natural resources research and management.

LITERATURE CITED

Dale, V. H., Doyle, W.T. and Shugart, H. H., 1985: A comparison of tree growth models. Ecol. Modelling, 29:145-169.

Goldberg, A. and Robson, D., 1989: Smalltalk-80: The Language. 585 pp., Addison-Wesley, Reading, MA.

Holland, J. H., 1986: Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems. pp. 593-623 in: Machine Learning: An Artificial Intelligence Approach, Volume II, R.S. Michalski, J.G. Carbonell and T.M. Mitchell (editors), Morgan Kaufmann, Los Altos, California, 738 pp.

Isebrands, J. G., Rauscher, H. M., Crow, T. R., and Dickson, D.J., 1990: Whole tree growth process models based on structural-functional relationships. pp. 96-112 in: Process Modeling of Forest Growth Responses to Environmental Stress, R.K. Dixon, R.S. Meldahl, G.A. Ruark and W.G. Warren, editors, Timber Press, Portland, Oregon.

Olson, R.L. and Wagner, T. L.. 1992. WHIMS, a knowledge-based system for cotton pest management. AI Applications ,6(1): 41-58.

Olson, R. L., Sharpe, P.J.H., and Wu, H., 1985: Whole-plant modeling: A continuous-time Markov (CTM) approach. Ecol. Modelling 29:171-187.

Schmoldt, D. L., 1991. Simulation of plant physiological processes using fuzzy variables. AI Applications, 6:3-16.

Zadeh, L. A., 1965: Fuzzy sets. Information and Control, vol. 8, pp. 338353.

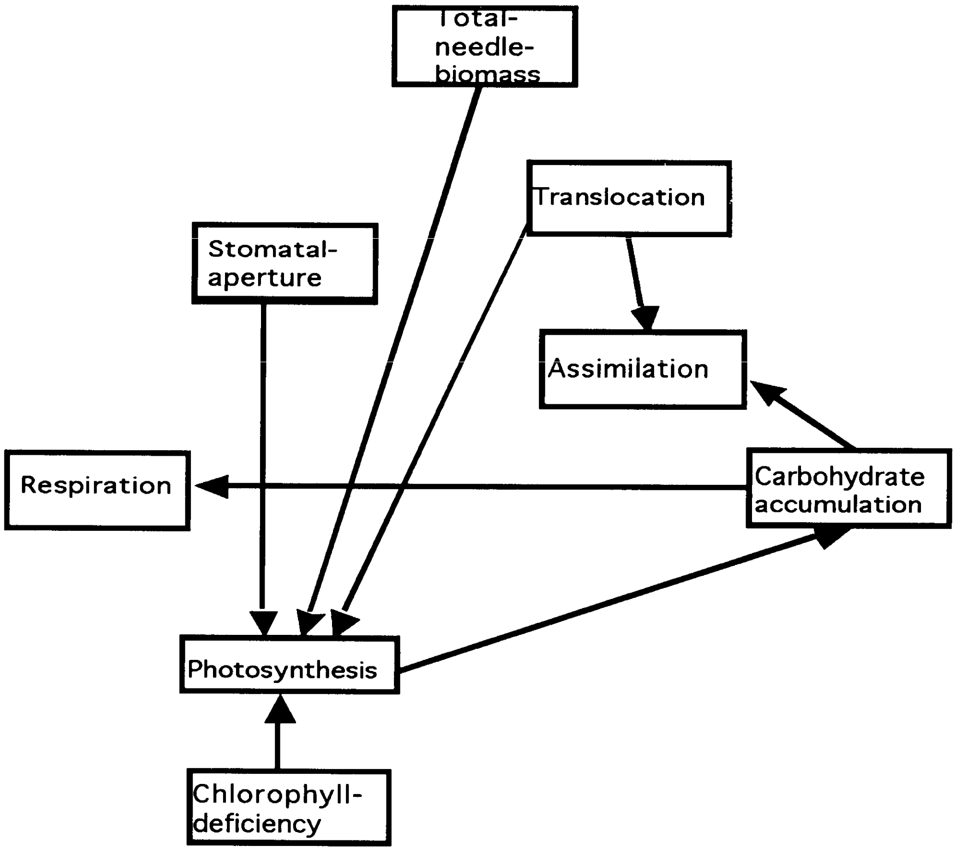
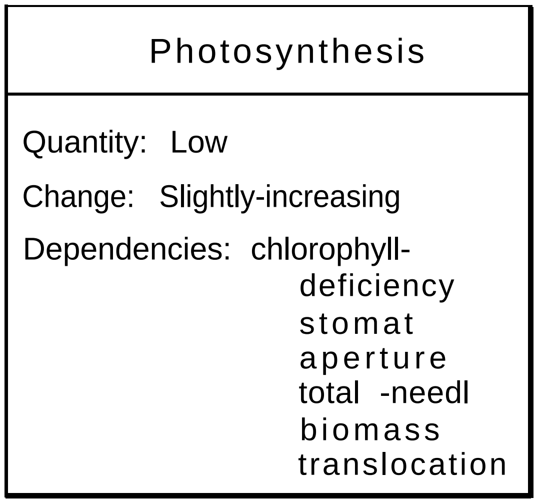


Figure 1. A fragment of the tree physiology model from Schmoldt (1991). See text for details.



Photosynthesis

Quantity: Low

Change: Slightly-increasing

Dependencies: chlorophyll-deficiency
stomat aperture
total -needl biomass
translocation

Figure 2. The structure of a parameter, Photosynthesis, from the tree physiology model of Schmoldt (1991). See text for details.

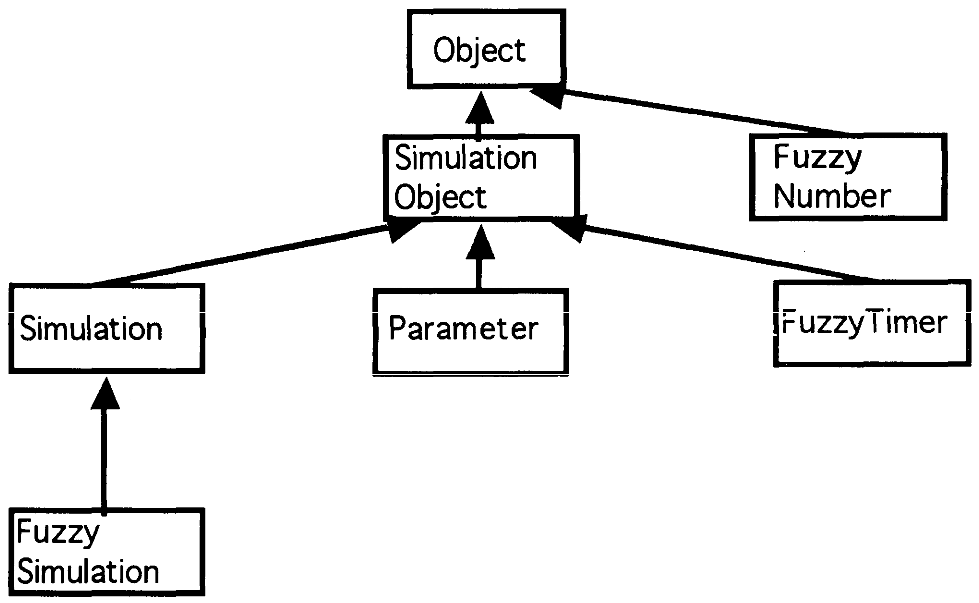


Figure 3. The inheritance hierarchy of Fuzzy-AF. See text for details

Class:	**Parameter**	
Superclass:	**SimulationObject**	
Data Structures:	quantity change influences dependencies timeScale	quantityLinguistic changeLinguistic influenceClasses dependencyClasses
Functions:	update updateQuantities	translate: with: updateChange

Figure 4. Definition of Class **Parameter.** See text for details.

Class:	**Fuzzy Number**
Superclass:	**Object**
Data Structure:	value
Functions:	+, -, *, /, >, exp distanceTo: interpolateBetween:and: makeConvex

Figure 5. Definition of Class **FuzzyNumber.** See text for details.

Class:	**Fuzzy Simulation**	
Superclass:	**Simulation**	
Data Structures	parts* changeDefinitions changeBasis partClasses*	timer quantityDefinitions quantityBasis
Functions:	initialize constructDependencies	constructNet constructInfluences

Figure 6. Definition of Class **Fuzzy Simulation.** See text for details.

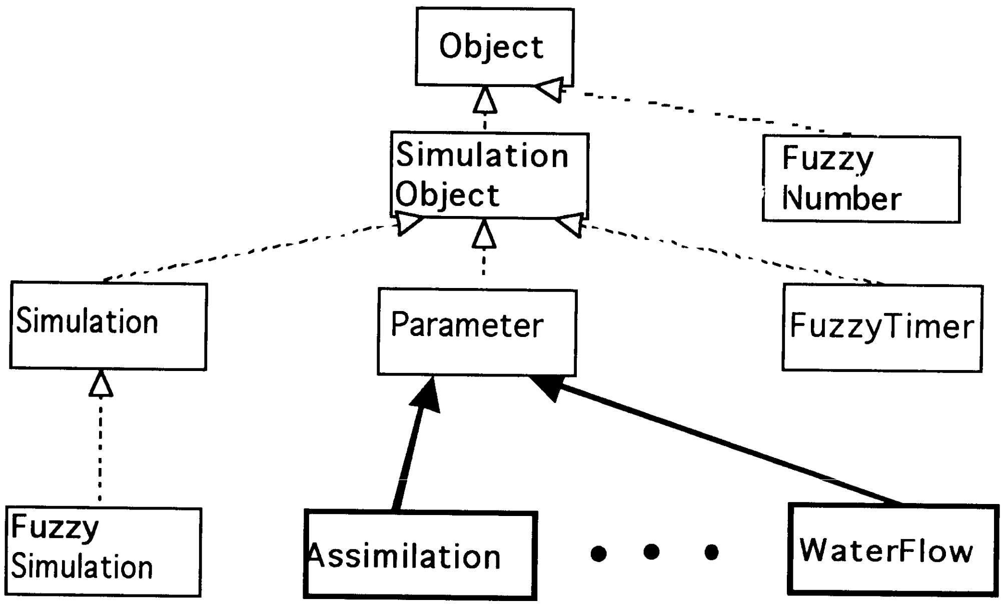


Figure 7. The inheritance hierarchy of Fuzzy-AF, with the subclasses for the tree simulation of Schmoldt (1991) emphasized. See text for details.

Caring for the Forest:
Research in a Changing World

Statistics, Mathematics and Computers

Proceedings of the Meeting of IUFRO S4.11-00 held at
IUFRO XX World Congress, 6–12 August 1995, Tampere, Finland

Editors
Michael Kohl
George Z. Gertner

Published by
Swiss Federal Institute for Forest, Snow and Landscape Research (WSL/FNP)
CH-8903 Birmensdorf, Switzerland

1996