Developing forest models in the Simile visual modelling environment

Robert Muetzelfeldt
Jasper Taylor

The University of Edinburgh
Institute of Ecology and Resource Management
Darwin Building
King's Buildings
Edinburgh EH9 3JU
Scotland, U.K.
{r.muetzelfeldt, jasper.taylor}@ed.ac.uk

Abstract

Simile is a visual modelling environment developed specifically for research modelling in ecology and related disciplines (www.ierm.ed.ac.uk/simile). It combines a System Dynamics (compartment-flow) with an object-based approach. The System Dynamics elements provide an intuitive language for modelling continuous processes, while the objects enable the modeller to handle various forms of disaggregation, spatial models, and individual-based modelling, with dynamically-varying population of individuals and interactions between them.

Simile thus provides the forest modeller with the ability to model the same stand of trees in quite different ways. The modeller may choose to have a lumped model, perhaps for looking at the carbon dynamics of the whole stand. Or the forest can be disaggregated into size, age, species and/or other classes. Or the modeller may choose to represent the stand as individual trees, with recruitment and mortality, and with competition between trees based on light, water or nutrients. And the stand model - however constructed - can be replicated spatially (in grid squares, hexagons or polygons) for landscape-level modelling.

The key point about Simile, however, from the point of view of the research community, is not its power as a modelling environment, but the fact that it conforms to a declarative modelling approach. Models are saved as structured text files, and these files can be processed by other software, developed by others totally independently of Simile itself. This software could include tools for presenting models in various ways, for querying the structure of models, for comparing two or more models, and for transforming models (e.g. from lumped to disaggregated or vice versa). This is illustrated with a program which generates an html description of any Simile model, complete with hyperlinks linking variables to the place where they are defined.

1. Introduction

The forest modeller faces many challenges. First, there are decisions to be made about how a particular stand of trees are to be represented: as pools of carbon or volume of timber; as the number of trees in different size classes; or as individual trees? Second,

there is frequently the need to broaden the model beyond just the trees, to include the other abiotic and biotic factors that interact with the trees. Third, there is the frequent requirement to include a spatial aspect: 3D space at the canopy level, and 2.5D space at the landscape level. Resolving these issues often leads to quite complex models, which are difficult to implement as conventional computer programs. Moreover, conventional implementations of such models make it difficult for others fully to understand the model structure; and re-use of submodels within the research community is low or non-existent. These problems with the current practice of modelling are well recognised (Acock and Reynolds, 1997).

Simile (pronounced 'similee') is a visual modelling environment that has been designed specifically to address these challenges and problems (Muetzelfeldt and Taylor, 1997a,b). It allows a wide range of different model types to be implemented within the same environment, including spatial and individual-based models. It provides a very user-friendly environment for model construction, thereby reducing the time and expertise required for constructing models. It greatly improves the ability of others to understand the structure of your model, through diagrams and clear listing of equations. And it supports modular modelling, opening the way for re-use of submodels within the research community.

In this paper we will first review the main features of Simile. This is followed by examples showing radically different ways in which the same forest stand can be modelled in Simile. We will then explore how other biotic and abiotic components can be incorporated into stand-level models, and how we can move from stand to landscape scale. We will then describe how models can be linked to existing data sets, and conclude by laying out some perspectives for future development.

2. Simile features

Visual modelling
Simile supports a two-phase approach to model construction. The first phase involves the drawing of diagrams that show the main features of the model. The second phase involves fleshing-out the model-diagram elements with quantitative information: values and equations.

System Dynamics
Simile allows models to be formulated in System Dynamics terms: that is, as compartments (stocks, levels) whose values are governed by flows in and flows out. This can be considered as a visual language for representing differential-equation models, with a compartment representing a state variable, and the rate-of-change being the net sum of inflows minus outflows.

Disaggregation
Simile allows the modeller to express many forms of disaggregation: e.g. age/ size/ sex/ species classes. This is done by defining how one class behaves, then specifying that there are many such classes.

Individual-based modelling
Simile allows a population of objects to be modelled. As with disaggregation, you define how one member behaves, then specify that there are many such members. In this case, however, the model designer can add in symbols denoting the rules for creating new members of the population, and killing off existing members. Individual members of the population can interact with others.

Spatial modelling
Spatial modelling, in Simile, is simply a special form of disaggregation. One spatial unit (grid square, hexagon, polygon...) is modelled, then many such units are specified. Each spatial unit can be given spatial attributes (area, location), and the proximity of one unit to another can be represented.

Modular modelling
Simile allows any model to be inserted as a submodel into another model. Having done this, the modeller can then manually make the links between variables in the two components (in the case where the submodel was not designed to plug into the main model); or links can be made automatically, giving a 'plug-and-play' capability. Conversely, any submodel can be extracted and run as a stand-alone model, greatly facilitating testing of the submodels of a complex model.

Efficient computation
Models can be run as compiled C++ programs. In many cases, these will run as fast as a hand-coded C++ program, enabling Simile to cope with complex models (100s equations; 1000s object instances).

Customisable output displays and input tools
Simile users can design and implement their own input/output procedures, independently of the Simile developers. In particular, you can develop displays for model output that are specific to your requirements. Once developed, these can be shared with others in your research community.

Declarative representation of model structure
A Simile model is saved in an open format as a text file (in Prolog syntax). This means that any one else can develop tools for processing Simile models in novel ways. For example, one group may develop a new way of reporting on model structure, while another may wish to undertake automatic comparison of the structure of two similar models. It also opens the way for the efficient of models across the internet (as XML files), and for the sharing of models between different modelling environments.

3. Alternative approaches to forest stand modelling

How should a single stand of trees be modelled? The answer clearly depends on one's objectives. But even for one objective ("the effective of thinning regime on yield"), different modellers will choose different approaches, reflecting personal preferences, the tools available, and so on. Simile is neutral as regards the question of 'best' approach: it supports a wide variety of approaches, and allows the modeller to choose one. This section gives examples of stand models reflecting radically different approaches, but all implemented in Simile's model -design environment.

Lumped carbon/biomass models
Some models treat the stand as a store of some substance (carbon, biomass, energy) on a unit of land, and model the flows in and out of this store. Fig. 1 shows an example based on a published model: the forest component of the McMurtrie and Wolf (1983) agroforestry model, in which the stand is represented as a store of biomass with increment and loss flows, and a network of physiologically-related influences.

Standard variations on this theme include the use of more compartments, representing the storage of the substance in various storages (leaf, stem, roots); and the inclusion of multiple substances (e.g. nitrogen, water), modelled by separate compartments and flow paths.

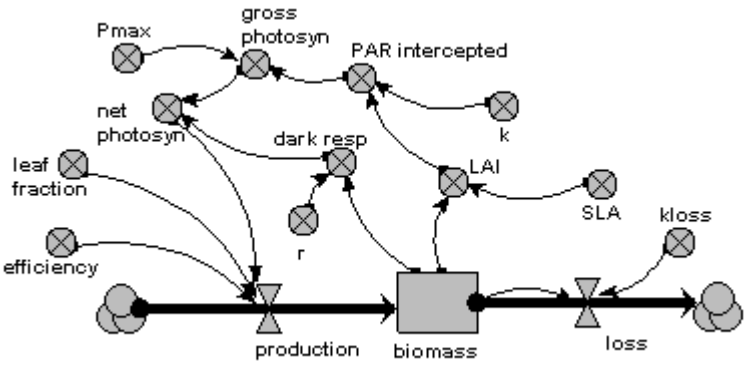

Fig. 1. Physiologically-based forest biomass model.

Size-class models

A common approach for forest management is to model the stand in terms of the number of trees in various size classes. Tree growth then translates into the rate of transition between one class and the next. Fig. 2 shows a re-implementation in Simile of Alder's (1995) stand-projection model for *Triplochiton scleroxylon*. Each compartment represents the number of trees in one size class, and there are flows representing recruitment, mortality, and the movement of trees between classes (growth).

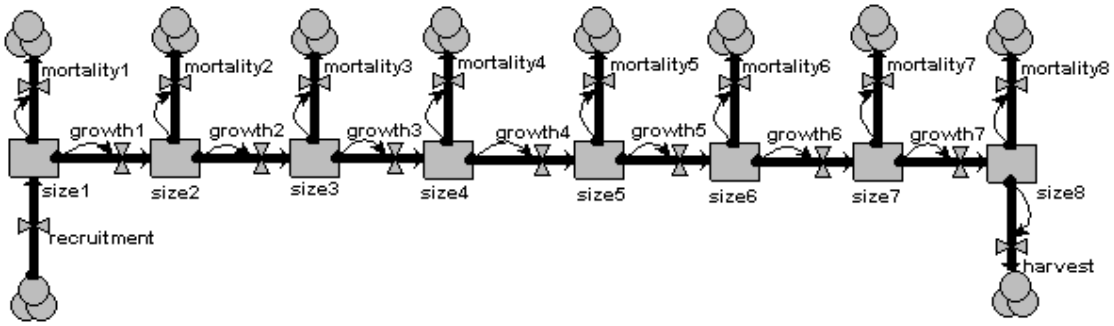

Fig. 2 Size-class model, using a separate compartment for each class

Fig. 3 shows the same model implemented using a multiple-instance submodel. Now, we just model one class, wrapped up as a Simile submodel specified as having 8 instances. Mathematically, this model behaves exactly the same; but clearly it is much easier to build and manage this model (e.g. change the mortality equation for all classes); and the model could be easily changed to (say) 50 classes.

Standard variations include modelling the stand in terms of age rather than size classes; the addition of stand density feedbacks; and the modelling of multiple species in parallel.

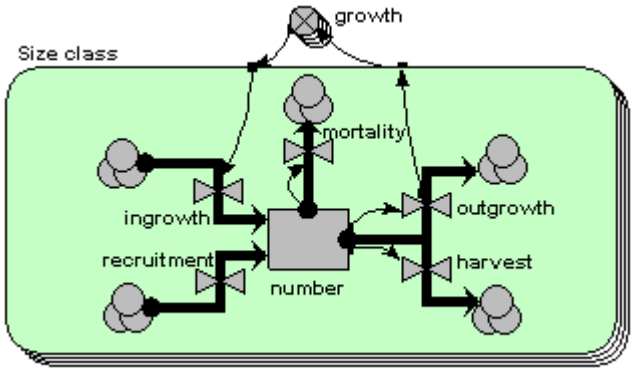

Fig. 3 Size-class model, using a multiple-instance submodel

Individual-tree stand models

The same stand could be modelled by representing each tree individually, with or without spatial location (distance-dependent or distance-independent models). Again, we make use of a submodel to represent one unit (Fig. 4), but this time it represents one tree rather than one size class. Each tree has a single compartment representing its size, and symbols representing the initial number in the stand (the sunrise symbol), the recruitment of new trees (the bird symbol), and the rule for killing trees off (the cross symbol). Each tree has location in space (x,y attributes), and this plus tree size is used to calculate a distance-dependent competition index.

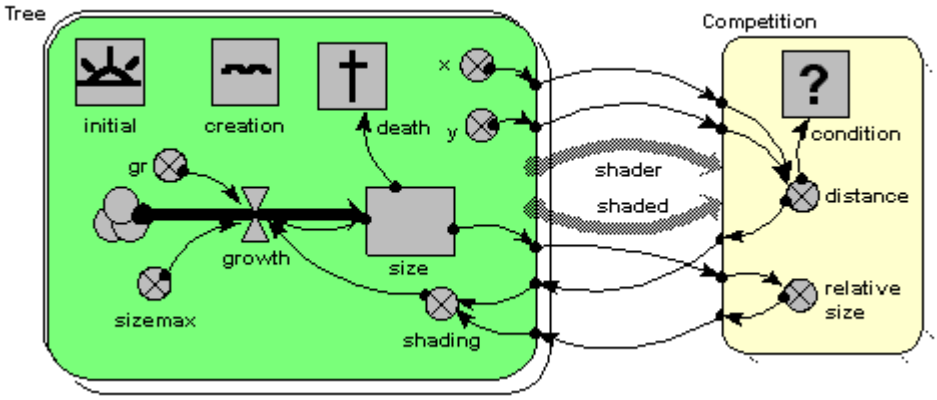

Fig. 4. Individual tree population model with competition between nearby trees

The model could be easily extended to include the species of each tree (simply add a new attribute for species) which could influence its growth and population parameters. The simple, one-compartment growth model could be replaced with a multi-compartment model representing carbon or biomass storages in tree leaf, stem and roots.

4. From forest-stand to forest-ecosystem modelling

Any of the above forest-stand models can be wrapped up as a Simile submodel, and form part of a larger, forest-ecosystem models. In fact, provided that they interfaced with the rest of the model using the same variables, they could be swapped in and out on a plug-and-play basis.

Fig. 5 illustrates this with a simple example. In this case, the stand is modelled using the McMurtrie and Wolf biomass model. The growth of the stand is influenced by soil water; and the stand biomass influences the growth of the herbaceous layer, which in turn is subject to herbivory. Water loss through transpiration is influenced both by tree biomass and the biomass of the herbaceous layer.

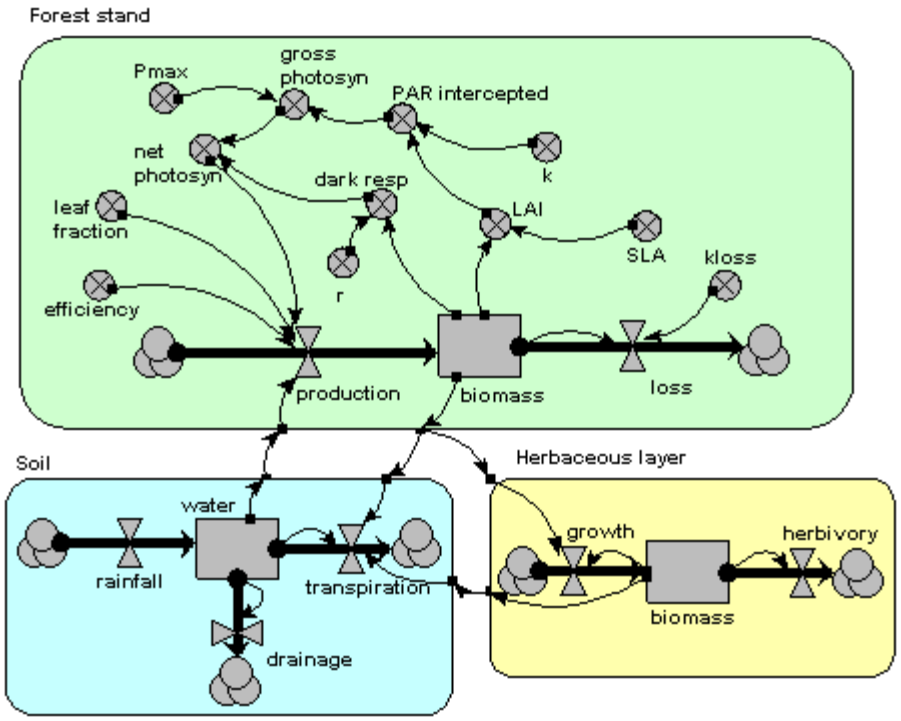

Fig. 5. Simple forest ecosystem model

5. Spatial modelling

Any of the above aspatial models can easily be made into a spatial model. All that we need to do is to wrap the model in another submodel envelope, specify that it has multiple instances (each instance representing one spatial unit), and give it attributes representing the area and the location (x,y co-ordinates) of each unit.

Fig. 6 illustrates this with the forest-ecosystem model introduced above. In this case, the area for each unit is set to the same value, and the x and y attributes are generated (in the equation for each variable) according to a regular pattern, so that every instance of the submodel represents one square on a grid. Alternatively, we could have chosen to model the spatial units as polygons, in which case the area and location of each polygon would probably have been read in from a GIS-generated file.

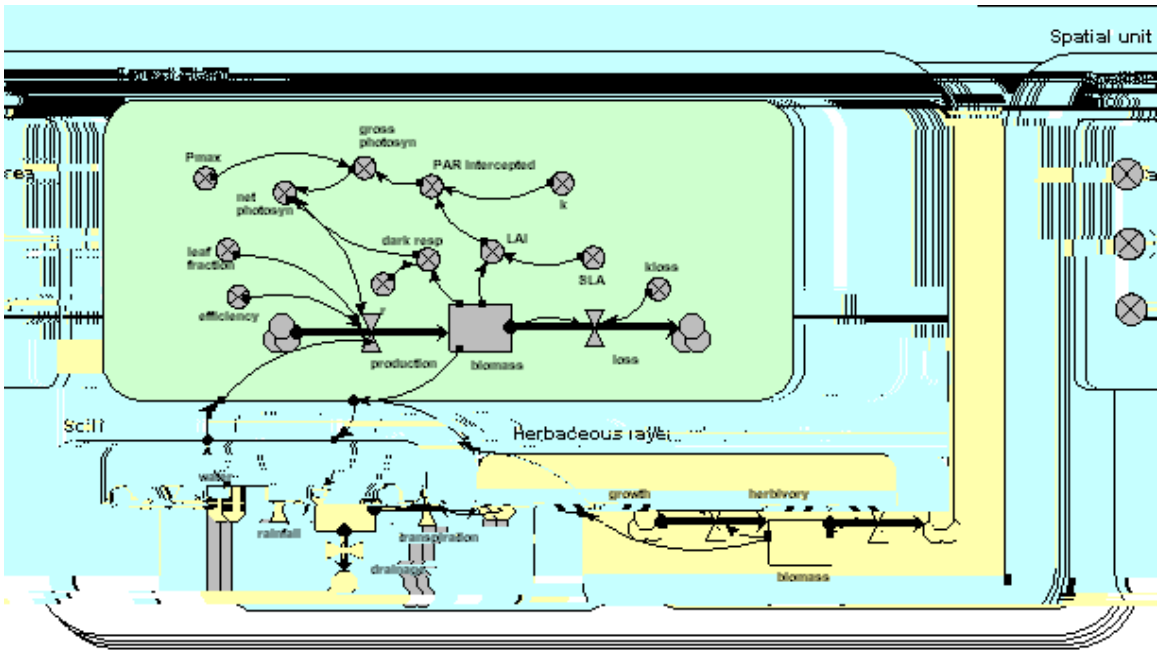

Fig. 6. Spatial version of the ecosystem model.
The previous ecosystem model is wrapped in a submodel ('Spatial unit'), and extra variables are added specifying the attributes of each spatial unit.

It is important to note that, although Simile can easily handle spatial modelling, it has no spatial-modelling concepts hard-wired into it. All we have done is to use Simile's normal model-design elements (the multiple-instance submodel and attribute variables) to represent information that we interpret spatially. Simile itself does not interpret 'x' and 'y' as spatial co-ordinates: they are simply variables with values. It is up to us to engineer them to correctly represent our spatial perspective. The grid display is certainly a spatial display: but it is not hard-wired into Simile. It is merely an example of a display added to Simile's list of display tools that happens to be suitable for showing a spatial grid.

6. Provision of data

Simile allows the modeller to separate the structure of the model from the data values needed to configure the model. The latter typically include:

- parameter values: e.g. a species-specific mortality rate;
- scenario values: e.g. permanent-sample-plot (PSP) data on tree species, height and location.

Simile caters for this requirement by allowing a file containing the relevant information to be associated with a model at run time. This file is called an SPF (Simile Parameter File). The SPF file can contain either actual values, or a reference to a column in a CSV (comma-separated value) text file. (CSV files can be readily exported from a spreadsheet package such as Excel.) In order for Simile to know that a particular variable will be getting its value from this file, the variable must be tagged as being of a certain type: a "file parameter".

This file is a text file, so it can be copied and edited by hand if you want to change inputs or set up different scenarios without having to go into Simile to do this. An

example is shown in Fig. 7, for a (hypothetical) individual-tree stand model. This SPF file contains two input values (mean rainfall and mean temperature), links to a CSV file containing GIS data for the variables in the Patch submodel representing the area, the x coordinate and the y coordinate of each spatial unit, and values for two parameters (gr and K) for each of 3 species.

inputs.spf

```
mean rainfall    = 975
mean temperature = 22.3

Patch/x          = GISdata.csv {x}
Patch/y          = GISdata.csv {y}
Patch/area       = GISdata.csv {area}

Species/gr       = 1 0.2  2 0.3   3 0.5
Species/K        = 1 12   2 13.5  3 9.1
```

GISdata.csv

```
ID,area,x,y,vegetation
1, 2.2, 34, 54, 7
2, 3.2, 45, 32, 6
3, 4.1, 54, 57, 6
4, 1.7, 55, 61, 6
5, 0.9, 38, 65, 3
```

Fig. 7. An example Simile Parameter File (SPF), and the first 5 lines of the Comma-Separated Value (CSV) file referenced by the SPF.
Each line of the SPF contains the full name of the Simile variable (including any enclosing submodels), followed by:
- a single value;
- a reference to a column in a CSV file; or
- a set of values, with corresponding index.

This mechanism makes it very easy to re-use data files: the same data file can be referenced by several SPF files. It also makes it very easy for a user of the model to change scenarios: you simply select a new SPF file and run the model again.

7. Declarative modelling

When a Simile model is saved to file, the file contains everything you need to know about the model: its mathematical structure, its visual appearance, and any comments that you have added. This file does not contain a conventional program: that is generated by Simile only when you wish to run (simulate the behaviour of) the model. What it actually contains is a set of text statements, in Prolog syntax, one for each node (compartment, variable...) and arc (flow arrow, influence arrow...) in the model.

This approach is known as declarative modelling. What we store is a specification of the model, enabling the model to be reconstituted at some time in the future. It contrasts starkly with the alternative (and far more common) approach, in which the primary (usually only) way in which a model is stored on a computer is as a computer program in a procedural (imperative) programming language such as Fortran or C.

It's inevitable that visual modelling packages will adopt a declarative modelling approach: there is no other way that they can store a representation of a model to allow the model developer to carry on working the next day. In some cases, however, the model is stored in a proprietary, binary format: no one else can know the format. In Simile, the format is open, so anyone else can write programs capable of processing saved Simile models.

An example of this is a program for generating an html description of any Simile model. This program is written in Prolog, and is totally independent of Simile itself. It generates a complete description of a model, submodel by nested submodel, printing out all available information on compartments, flows, variables and other model elements in a an attractive readable format. Moreover, all model elements referred to in one part are hyperlinked. Seeing the flows for a compartment, you can click on any one and be taken to the equation for it. Or you can click on any variable in an equation to see where that variable itself is defined.

8. Future development

It's easy to think of Simile as a software product that does a certain job - or rather, a suite of jobs - related to modelling and simulation: building a model, displaying its structure, simulating its behaviour, etc. In this view, future development means improving the way that Simile does these jobs, and increasing the number of jobs it can do.

This is not the way we see things. Certainly, we will be continuing to develop Simile, and we have a big long list of enhancements that we plan to introduce. But rather than think in terms of "a software package that does some jobs", we prefer to think in terms of a distributed pool of models and tools for handling models. In this view, tools for handling models can be developed by a wide range of groups: what they share is the ability to work with any model in the pool of models. Simile is then just one of the tools that will be around in a few years time. Other tools may do other tasks (such as automatic comparison of the structure of two models). Or indeed they may do the same task as Simile, setting up in competition.

This view matches precisely the current vision for e-science. 'E-science' is science done over the internet. It aims to create a world where scientists can access the resources (data and software) they need for doing their science, without having to worry about where these resources are located. The technology for doing this is generically called The Grid (by analogy with a power grid: when you plug a kettle into the mains, you don't have to know which power station supplied the electricity). It is the subject of a huge investment in Europe and the USA - £118 million over 3 years in the UK alone - though almost exclusively restricted to Big Science: nuclear physics, biotechnology, remote sensing.

In this scenario, how should models be represented? The prevailing view within forest modelling in particular, and ecological/environmental modelling in general, would be that models should be represented as computer programs: in Fortran, C++ or whatever. In that case, obtaining Grid-style functionality requires the ability for models to talk to each other, perhaps through COM (Common Object Model) or CORBA (Common Object Resource Brokering Architecture) technologies. In our view, a declarative modelling approach, as outlined above, is far superior. By separating the design of a model from its implementation in a runnable form, it opens the way for tools to assist in the model-design process, allows a wide range of tools to be developed for performing other useful tasks with models (like interrogating their structure), allows the same model to be simulated on a wide variety of platforms, and - most crucially - provides a far

greater guarantee of the long term persistence of the model, so that people will be able to recover it even if the tools for handling it no longer exist.

9. Conclusions

Simile is an advanced visual modelling environment that is capable of handling most of the types of models used in forest modelling, and of generating computationally-efficient simulations. Because it is based on a declarative modelling approach, it be seen as just one tool of many that could exist in the future, as others develop their own tools for handling Simile models.

Simile's main role is to act as a proof-of-concept. For many years, many groups have been bemoaning the lack of tools for building models and the lack of shareability of models and submodels. A major obstacle for the community as a whole to actually do anything about this was the belief that the range of modelling requirements was too great, and that therefore it would not be possible to develop a common approach. Simile has gone a long way to disproving this. Therefore, we (the forest/ecology modelling community) should engage in a process of consultation and standards development, laying the foundation for cooperative modelling in the future. The emergence of e-science and The Grid concepts gives extra justification for such an initiative, and opens the door to totally new ways of doing science.

Acknowledgements

The development of Simile has been mainly funded by the United Kingdom Department for International Development (DFID) for the benefit of developing countries The views expressed are not necessarily those of DFID. Project codes R5652 and R7635 Forestry Research Programme. Some development has also been funded by the EU Environment programme, within the ModMED project We gratefully acknowledge the support we have received from these sources.

We would also like to thank the many users whose comments have led to substantial improvements in Simile's user interface.

References

Alder, D. (1995) Growth Modelling for Mixed Tropical Forests.. Oxford Forestry Institute, Dept of Plant Sciences, University of Oxford. Tropical Forestry Papers No. 30. 231 pp.

McMurtrie, R. and Wolf (1983) A model of competition between trees and grass for radiation, water and nutrients. *Annals of Botany* 52: 449-458

Muetzelfeldt, R.I. and Taylor, J. (1997a) The suitability of AME for agroforestry modelling. *Agroforestry Forum* 8(2): 7-9

Muetzelfeldt, R.I. and Taylor, J. (1997b) The Agroforestry Modelling Environment. In: Agroforestry Modelling and Research Coordination, Annual Report 1996-97, ODA Forestry Research Programme, Project R5652. NERC/ITE Edinburgh.

Reynolds, J. and Acock, B. (1997) Modularity and genericness in plant and ecosystem models. *Ecol. Modelling* 94: 7-16