A Report on the Studies of Possible Software Options for the Growth and Yield Modelling (GYM) Framework

Paul Phillips

12th November 1997

1 Introduction

This report follows work done on the evaluation of the use of various existing software for the purposes of fulfilling the objectives of the Growth and Yield Modelling (GYM) project. The project proposal (van Gardingen *et al.*, 1997) sets out clear targets for the project, which lead to objectives which must be born in mind when designing the software. These are stated in section 2.

The project proposal specifically mentions two existing software items (SYMFOR and Hybrid), however there are many other packages designed for similar purposes which should be considered for use (see section 17 (van Gardingen *et al.*, 1997)). Those considered at any level are listed in section 3. The majority of this report is given to the discussion and evaluation of the specific use of items of software within this project. Many of the packages were not written with this particular mode of use in mind, and, whilst being worthy tools in their own right, may not be the most suitable solution in this instance. The evaluations were made with specific reference to the requirements of the project as listed in section 2.

Finally, the author chooses between the possible options of ways forward, giving reasoning where appropriate. This can be used as a basis for discussion within the project team.

2 Requirements of the Framework Objectives

Target Institutions:

- National or regional research organisations (e.g., BPK- Samarinda, FRIM);
- CIFOR;
- Government officials (e.g., Ministry of Forestry, Policy);
- Forest managers (i.e., concession holders);
- Certification bodies (e.g., Lembaga Ekolabel Indonesia, Forestry Stewardship council); and
- Development projects (e.g., EU Berau Forest Management Project, DFID bilateral aid programme).

The framework should eventually be available to, and usable by, all of the above. This has implications for the appearance of the software to the end-user, and the ease of use of the software (taking into account the level of computer literacy and modelling competency necessary of the end-user in order to use the framework correctly). For example, the only person to actually create models using SYMFOR was its primary author, as contrasted with the large number of people who ran the models. This is acceptable if the models are valid for the data on which they are being used, but raises the question of whether a modelling pre-compilation framework was necessary or whether a simple model (with options of which module to use) would have sufficed.

Links to other ODA funded projects:

- This project will utilise the outputs from ITFMP.

The obvious output is SYMFOR. However, more general outputs include: a modular approach to managed tropical forest modelling; a user-friendly windows-based front end for use by forest managers (amongst others); recognition of the importance of models which run fast; the increased awareness of SYMFOR by forest managers; and lessons concerning the need for quality documentation, promotion and completion of exportable software products. Some if not all, of these outputs must be utilised in order to satisfy the project requirements.

The project's objectives:

- To predict forest growth and yield following management interventions using a modelling framework; and
- To evaluate criteria and indicators of ecological and economic sustainability.

The first of these requires both that growth and yield be modelled and that management interventions be modelled. These are two distinct tasks which affect the same data: growth can be described using empirical methods, a process-based approach, or some combination of the two; management interventions include choice of trees to log (individual trees, trees over a certain diameter or of a certain species, etc.), method of logging, skid-trail design, road placement and so on. The second objective requires that the completed framework be used to produce significant results before the end of the project. This places limiting requirements on the time-scale allowed for the completion of (at least a prototype of) the framework software, the relationship calibration and the overall validation of the models generated using the framework.

3 Modelling framework solution criteria

There are three sets of criteria which must be satisfied by any possible design for the GYM framework relating to the end user, in addition to the considerations outlined above:

- The framework must have an interaction with the user which allows the user to do what they want easily: a windows-based front end; to choose different management

options easily to enable comparison; a multiple-run facility; suitable graphics; and optimised run-speed.

- A model made using the framework must be able to run sensibly using existing data only (Rombouts, 1997).
- The framework must provide the outputs and flexibility of outputs, which the user requires, in the form required by the user (e.g., as inputs to SAS, Excel, etc.).

In addition to these essential criteria, several features have been identified as desirable for possible frameworks, as specified below.

- Documentation at all levels:
  (a) on-line documentation of the use of the framework, including calibration and validation,
  (b) documented code and a rulebook for simple or common operations, including the documentation process itself,
  (c) fully commented code.
- A code structure/development environment that is easy to maintain:
  (a) modern languages and/or development systems,
  (b) simple software structure where possible,
  (c) a minimisation of dependency on software supported externally to this project,
  (d) single-language programming where possible
- A move to better (and more modern) programming practice:
  (a) 32-bit programming,
  (b) 'OO' and "top-down step-wise" approach to programming.
- Flexibility of input and output styles, or at least facility for other style options to be added. Included in this is the possibility for using data other than PSP data, such as inventory data, and porting data to/from other data-handling packages (e.g., SAS, FoxPro).
- A "native" feel to the software for anyone familiar with PC software, so that they can be confident of how to use what is presented to them.
- Simplicity and logical nature of the use of the software (e.g., not too many menu options, not too many features presented at once).
- Software which allows the choice of modules to be made at run-time. This would bypass the need for a separate model-making section to the software, and would, in a sense, unify the modelling process.

The implementation of many of these features will make the development of the code slower than would otherwise be the case, but in my experience they make the code more robust and more likely to be used and maintained, and therefore represent an advantage.

4 Software considered

The following software was considered:
SYMFOR (Muetzelfeldt & Young, 1996),

Hybrid (Friend *et al.*, 1997),
AME (Lawson *et al.*, 1996),
RAFOM (Leersnijder, 1997),
DIPSIM (Ong & Kleine, 1995),
FORMIX (Huth, 1997),
BWIN 2.5 (Nagel, 1997),
SmartForest (Orland, 1997)

5 SYMFOR reviewed

SYMFOR was written to satisfy many criteria similar to those of the current GYM project, and it was originally intended that SYMFOR should form the basis of this project. It has many of the features necessary to satisfy the modelling framework criteria described above: a user-friendly and flexible front end, with appropriate graphical representations of the plot; it uses PSP data for initialisation; it runs fast; its outputs can be read in directly by Excel and SAS; it contains fully functioning individual tree models; the method of running SYMFOR is well documented and workshops on its use have been successfully run.

There are, however, some known deficiencies of SYMFOR that would need to be addressed if it were to be used for this project. The only environmental factor affecting growth is currently light: the proposal has been made to include some elements of Hybrid, such that soil nutrients and water availability also contribute to the growth calculation. The procedure for including alternative algorithms for the various processes is complicated and undocumented. The process of calibration of the procedures within SYMFOR is undocumented, and has not been done at all for most relationships. Last but not least, the current version of SYMFOR has bugs in: it crashes after only a few runs, will not compile on the latest compilers or windows NT software and has inconsistencies in the program design (such as using DDE communication where simple subroutine calling would have sufficed, and using a DLL for some fundamental routines but not for the modelling procedure functions).

There are myriad potential ways to take SYMFOR forward for the purposes of the GYM project, involving the restructuring of the code, rewriting of various sections of the code and adding functionality and/or graphics from other existing models. In this section, various parallel ideas are put forward and assessed on their practicality (for implementation), their purpose and their importance to the project.

- Add model manager options to choose between modules to use for a given functionality (e.g., growth). Currently the modules must be chosen beforehand and compiled into an executable. In practice, it has been found that end-users of the system do not find this easy and so don't do it. It would be made more approachable to them if there were simple menus to select which routines to use at run time. Also, there would be no need for a directory structure based on the different models as all models would exist in one executable.

- Remove the module structure by putting all modules into the DLL. This would reduce the framework to just the model manager and a DLL. The DLL routines would be called by each other and by model manager routines. This simplifies the structure of the code without losing any functionality.
- It will be possible to make the code object-oriented. This highlights the process-based nature of the model, and enables a clear understanding of the flow of the data within the model. It also leads to running speed improvements through a programming style more optimised for C++ compilation.
- Remove the DDE communication links by using direct calls to subroutines. DDE links may be used for communication between the framework and other Windows applications, but they are not necessary within a single application.
- Remove the Visual Basic model manager by re-writing it in C++, such that all the code is in the same language. For a discussion on the choice of language, see section 12. It is suggested that the model manager should be a rather separate software system to the object-oriented DLL modules, since it does not represent the same system. It would, however, initialise and control the execution of the modules.
- Add a Hybrid tree object subroutine and environmental factor subroutines (which will return constants initially except for the competition for light). Initially this will not necessarily give a better estimate of growth than from current SYMFOR modules, but the possibility of adding the effects of local climate, slope, soil type and water flow give the chance to estimate growth and yield in areas where previous growth data is not available.
- A choice of method for deciding on the amount of light reaching each tree remains to be made, since the current SYMFOR one is very simplistic, and the current Hybrid does not calculate it individually for each tree, as it is a gap model. It is proposed that the SYMFOR method be used initially.

Considerations for the target institutions:

(a) The old-style front-end would be re-written, but done so as to resemble the current version of SYMFOR. There are no identified problems with the usage of the current front-end, so there is no reason to re-design it. This maintains the user's familiarity with the software appearance.

(b) The choice of which modules to use would be saved as named sets (e.g., model60 or model63) which could be selected, or existing choices could be edited. Since all the modules would be stored in the DLL, any module may call any subsidiary module without the subsidiary having been named by the user, thus no knowledge of module-dependency is required by the user. Suitable rules would be coded to prevent duplication of routines (e.g., two separate growth routines being selected), and the choices would simply set flags for the model manager to decide which module to use. This should be a substantial improvement on the current SYMFOR version, and should allow, for example, comparisons of logging strategies to be compared in consecutive runs without requiring a new model to be compiled. This facility would be used by all target institutions.

(c) The Hybrid tree object may reduce the speed of execution of the code, due to the nature of the equations being solved, however this is difficult to determine without testing, and is a natural consequence of introducing process-based modelling. If it

runs significantly slower it may be an inconvenience to end-users, or may act as a deterrent to its use.

Considerations for the output from ITFMP:

(a) All of the outputs from ITFMP would be used in this framework, including documentation and knowledge bases from all institutions.

Considerations for the project's objectives:

(a) Individual tree modelling will allow all logging strategy types to be modelled. With the choice of modules being made at run-time, comparisons of growth and yield results for different management interventions will be simplified. The introduction of the Hybrid tree object and associated environmental dependency allows more realistic modelling of growth, and more confidence in modelling areas which do not have previous growth data available.

(b) The only programming necessary for a basic version of this model is the model manager, template environmental influences routines and some management options. The author estimates that this could be available in as little as 8 months. The debugging and testing would then be coincident with the on-going development of more sophisticated routines. The most recent version could then be used for evaluating criteria and indicators. The modularity of this framework would allow continuous development. Very little of the existing SYMFOR would need to be discarded, although much could be improved and may be re-written in the process.

6 Hybrid reviewed

Hybrid (Friend *et al.*, 1997) is a process-based model designed to simulate the major physiological processes involved with plant growth, in order to evaluate interactions with the environment. Hybrid has not previously been used as a part of a growth and yield model, and has no facility for implementing forest management decisions. The process-based growth modelling and the consequent environmental effect on growth is of relevance for three reasons:

- evaluating yield in a region where previous growth data are not available,
- evaluating yield in a region with a non-constant climate, particularly with respect to drought, and
- evaluating indicators of sustainability for managed tropical forests.

Hybrid models individual trees, but is a “gap” model; over one plot, within Hybrid, there is no horizontal disaggregation.

Two methods have been suggested for implementing process-based growth into a framework for modelling managed tropical forests:

1. Use the understanding of the processes developed by Hybrid to generate a “tree object”. Each tree object in a model would represent a tree in a plot, and each simulation time-step the model would grow the tree, using data from routines modelling the environment (light, nutrients and water). Such a program would resemble the current SYMFOR more than Hybrid, and so the suggestion is to put a

"tree object" into a SYMFOR-like system, adding environmental quantities as necessary. This concept was discussed in the "SYMFOR Reviewed" section.

2. Use Hybrid as it is, adding a model manager for run control and some routines for the forest management options. This maintains the structure of Hybrid, whilst increasing its functionality. Visual C++ would be used to produce a user-environment similar to that of the current SYMFOR, with similar functionality. The choice of language is discussed in section 12. There is no suggestion to re-write the entire Hybrid code in a different language, as it would be an advantage of the system to have the Fortran code maintained along with the standard (ITE) version. In order to incorporate graphics, the forest management options would also be written in C++, which would be called from the existing Fortran routines. These would edit the tree data, record the logging actions and return to the Hybrid growth routines. The choice of logging type would be set in the model manager, and the logging itself may or may not be interactive. The model manager would read in data from the database files or elsewhere, and would set the output options. The data required for Hybrid would be generated using the databases and using the data on categorisation of tree species, as developed by G. Lawson. Figure 1 shows the suggested structure of this framework.

Considerations for the target institutions:

(a) It would look like SYMFOR, by the design of the C++ windows, so that current SYMFOR users would simply see additions to the current version.
(b) End-users would not have to re-compile the modules in order to make a model: the model manager would present any modules available as options.
(c) The model may run slower using process-based growth, though this cannot be properly quantified until a working version is produced. If it runs significantly slower it may be an inconvenience to end-users, or may act as a deterrent to its use.

Considerations for the output from ITFMP:

(a) With appropriate planning of the new model manager, all of the outputs from ITFMP could be maintained.

Considerations for the project's purpose:

(a) The framework would not be completely modular: either all or none of the Hybrid routines would be chosen. Thus it would not be possible to compare different growth modelling techniques, for example, and this may limit comparisons with calculations made by older versions. It would also make it harder to add other facilities as the framework develops.
(b) Since the only programming required is for the model manager and the management options, the time taken for this option to have a first working version could be as short as 6 months. Subsequent calibration and testing of the model might take longer, however, due to the lack of options within Hybrid for using alternative routines (e.g., for the environmental influences).

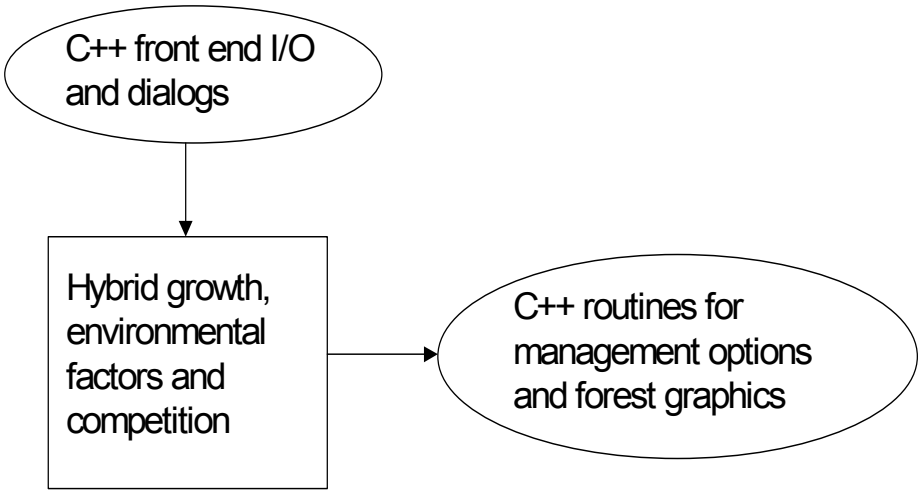

Figure 1: Showing the suggested structure of a framework incorporating the whole of Hybrid. The ellipses represent code that has yet to be written, and the rectangle represents code that requires only minor modifications.

7 AME reviewed

AME (Lawson *et al.*, 1996) is designed for producing models, sub-models and System Dynamics logic. Model visualisation (of the System Dynamics and sub-model structures) is a strong point. It is simple to load old models and modify them. It is possible to run command-prompt executables within the program, by calling them using a "TCL helper application" (TCL is a scripting language written in C). Models can be exported in several ways: as a specification in prolog from the visual representation, as C code that can be compiled (with the TCL source code, enabling use of the TCL helper applications) and run, or as TCL. All non-diagramatically represented code is written in TCL as helper applications, including I/O, dialog boxes, graphics, variable setting and run-control. TCL and C are extremely portable, so the whole code including the graphics can be used on many platforms. AME is still in it's development stages; for example, parameter values can only be changed by clicking on the relevant item in the system dynamics diagram rather than by name in a dialog box. This, and other issues which have still to be uncovered, would be resolved along with the development of a GYM framework.

Possible roles for AME in the GYM project:

1. Using AME as the modelling framework.
2. Using AME to produce executable models, and using TCL helpers and a TCL model manager to constitute the modelling framework.
3. Using AME to produce the prolog specification and C code for use in a SYMFOR like similar to that that is written by hand for SYMFOR currently.

7.1 Using AME as a modelling framework.

AME is a modelling environment (for a definition of the way this term is used in this text, see Appendix A). It cannot currently function as a framework (e.g., SYMFOR) because appropriate models do not yet exist within AME. In addition, the helper applications, allowing data input and output and choices of initialisation data file and parameter values to be made would need to be written. Running a model would involve starting AME, opening a model, clicking on "Run" and then following the instructions from the helper applications about which files to use, what data to output and maybe in what form, how many simulated years to run for and so on, then letting the model run. Running the model again would involve clicking "Run" again, and again inputting similar information. This mode of use is not seriously considered for use because of the run-time involved. The program runs interactively in TCL, which, being an interpreted language, runs much slower than equivalent code written in a compiled language. This would in itself inhibit the use of AME in this manner within GYM.

7.2 Using AME to produce executable models, and using TCL helpers and a TCL model manager to produce the modelling framework.

This idea would be to use AME model visualisation to develop and test models, then export them as C code for serious use. Users who would not develop their own models would probably still want to change parameter values, choose initialisation files, etc., so facilities would have to be in place to allow this. These facilities should be the same as are used by the model developer for integrated development, so TCL helper applications would be written for this purpose. These would be used whenever a GYM-type model is being produced. The models produced would be prototypes, with no initialisation data and default parameter values. A model manager (written in TCL for compatibility) would run the program in place of the current AME run-controller. This would have options of which model program to run, which initialisation file to use, which parameter set to use (although this is model-specific) and other multiple-run options (actually very similar to the way the SYMFOR model manager works currently). The TCL helper applications, which appear when the program is running, would follow the instructions from the model manager, controlling the actual input and output of data and the flow of data to and from any external sub-program, such as one from Hybrid modelling tree growth. The architecture of such a system is shown in figure 2. Methods of data I/O to AME models would need to be created, as the only method currently available is typing in values by hand.

Considerations for the target institutions:

(a) This could look like SYMFOR (small differences with using TCL rather than Microsoft windows to produce the graphics), so end-users could handle the transition smoothly.

(b) This system would require the same amount of modelling skill as using SYMFOR, as the system dynamics would only be used by those designing models, rather than those using them (a "model" in this sense is purely the structure of equations

governing relationships and the flow between relationships, not including parameter values or initialisation data).

Considerations for the output from ITFMP:

(a) All the benefits of work on SYMFOR, except the code itself, would be utilised. The TCL helper applications could be designed such that users who do not expect to design models could use this system as they would SYMFOR.

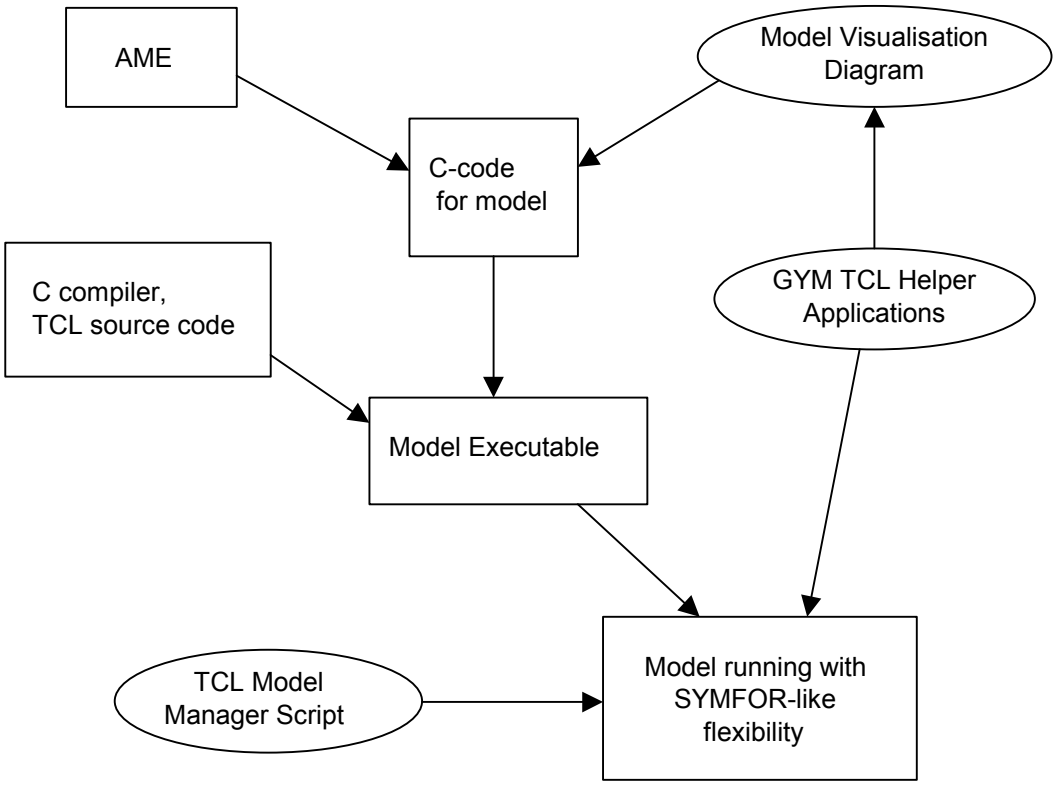

Figure 2: Showing the architecture of a possible framework based on AME. The rectangular boxes represent code that either already exists or can be created with minimal effort. The ellipses represent significant quantities of programming or design that are required before the system can function. The GYM TCL helper applications enable the user to create a GYM-like model visualisation diagram in AME. AME can then use the information from the diagram to export the model as C code, which can be compiled with the TCL source code to create a model executable. This could run under the command of a model manager, and using the same TCL helper applications as were used when the mode was designed.

Considerations for the project's purpose:

(a) Any comparison between modules (e.g., different possible management options) would have to be done by creating a new model for each choice of options. For example, for two possible growth sub-models and three different felling strategies a total of 6 models would be required. Each of these would have to be created

separately, and regenerated if, for example, a new TCL helper application was added to enhance performance in some way.

(b) All sub-models, models, the front-end (helper applications) and the procedure for linking all these together would have to be designed and written from scratch. This is in addition to other improvements to AME which are identified during development of this project. These tasks are not something that have been done to any great extent before and so it is difficult to estimate the time involved before a working version is ready for use. My personal estimate is a minimum of 1 year, with most time being spent.

(c) A Hybrid module to calculate tree growth could be written as a system dynamics sub-model within AME, since it is not nearly as complicated as the entire Hybrid code (see the previous section, "Hybrid reviewed"). Alternatively, if the module were already written in C, it could be added as code to be compiled along with AME-exported C code. This would run significantly faster than the standard method of using an executable tree growth module called by TCL, although it would need to be inserted manually for each model exported from AME.

7.3 Using AME to produce the prolog code similar to that that is written by hand for SYMFOR currently.

AME could be easily modified to produce a prolog output specification of the form currently used by SYMFOR, and the C code to go with it. The C code could be inserted into a standard SYMFOR shell, containing the communicating 'DDE' links, and could then be compiled and run using a SYMFOR model manager. This would, in effect, replace the unused model-designing methods of SYMFOR with AME's model visualisation. There is a problem with this approach, however. Model designers would not be able to test and run their models using AME in the same way that the model-users would under SYMFOR. TCL helper applications could be written to allow the AME running environment to behave in similar fashion to that of the SYMFOR model manager, but if a new facility were added it would need to be written separately for AME (in TCL) and SYMFOR (in Visual Basic). This is considered to be an unsustainable feature, and so this option is not taken further.

8 DIPSIM, reviewed

DIPSIM (Ong & Kleine, 1995) is an output of the Malaysian-German Sustainable Forest Management Project for Sabah. It is an empirical individual-tree growth simulation model, designed to be used as a forest management planning tool for Sabah's commercial Mixed Dipterocarp Forests. It has the following features:

- Site quality index: "good" or "poor". This is dependent on the "soil association", which derives from the "parent material" (rock-type), and "landform" (e.g., "low hills and valley, slopes less than 15°").

- Growth: basal area increment calculated from the species (20 species groups), tree size, site quality and competition. Competition is evaluated using the basal area of other trees in the same 20m by 20m subplot (no individual positional data is used).
- Mortality: the probability function is calculated from diameter, basal area and basal area competition. 7 species groups were parameterised separately.
- Recruitment: the probability function is dependent on the "stand density class". 7 species groups are identified, as there were not enough data for more.
- Harvesting: several factors constitute a stocking assessment, which, if passed, leads to the user entering the number of trees to be felled. These are not selected manually, and use a fixed minimum diameter of 60cm. A damage factor is entered by the user.
- Non-flexible outputs: output in database format is made every 5 simulated years, in fixed-structure data tables.

No use of the DIPSIM project is proposed here, other than to gain experience from it:

1. The same data should not be used for validation as for calibration, or it can be considered that the model is not validated.
2. Catastrophic events (e.g., the drought due to El-Niño) "appear to have a significant influence on model predictions". This is a statement implying that modelling the environmental factors affecting growth is worthwhile.
3. It has a look-up table for the functional groups data, which can be browsed using a model manager option.
4. It can use medium-term inventory data as well as PSP data. This is something which SYMFOR end-users have asked for recently.

9 FORMIX, reviewed

FORMIX3 (Huth, 1997), is a process-based gap model designed to model the growth of the managed mixed Dipterocarp forests of Malaysia. It uses 5 species groups, which differentiate between growth characteristics and their typical mature height. It is used to evaluate the biomass and numbers of trees in 5 distinct canopy layers. In each layer, it models the processes of photoproduction, respiration, the shading of lower layers and the growth of trees between canopy layers. Mortality is treated stochastically. Recruitment is assumed to be uniform in space and time. Silvicultural treatments and logging strategies can be modelled. The data required is inventory data, with some physiological data for each tree type in each forest type (such as lowland dipterocarp forests) for calibration. They appear to have access to this data from Malaysian lowlands, at least, and these values can be used (within appropriate limits) for other regions of tropical forest for those tree types. The model has been applied in three areas with different site conditions. At present the model is limited to evergreen rainforests, it cannot model droughts and assumes a constant climate. A soil/site properties module is currently under development.

This model appears to be similar to the proposal for this project of using Hybrid with a graphical front-end and management options. The processes it models appear to be similar to those modelled by Hybrid, although more detailed comparisons are not possible with the literature available at present. It does not account for environmental

factors other than light, at present, although the significance of this depends in any case on the nature and quality of the data available for those regions. It appears that a working version of a Hybrid growth and yield model would be very similar to the completed FORMIX project. No time-scale is given for completing FORMIX, however it has been developed over at least 6 years so far. For the introduction of physiological processes to forest modelling for Indonesia, the data FORMIX has used for its physiological parameters is of significant potential use.

A potential software solution for this project could be to use FORMIX, writing new data interfaces and maybe output routines. The outputs from ITFMP would be lost, unless a graphical front-end similar to that of SYMFOR could be used. The combined skills and application of the Kassel group with the DFID GYM project would represent a significant improvement from either individually, but, as is common in such collaborations, the differences in ambitions of the parties could slow progress considerably. No communication with the Kassel group has been made with respect to this suggestion.

10 RAFOM, reviewed

RAFOM (Leersnijder, 1997) itself is a sample program. It was written to develop the author's knowledge of how models of tropical rainforests could be constructed. There are some particularly useful features of RAFOM which could either be bought from the Forest Graphics company or written specifically for this GYM project, relating to graphical interaction with the simulated forest. The forest can be viewed schematically (using "lollipop" style diagrams) either from the side, the top or in a projected view. From any of these views, the user may click on individual trees to display data about that tree or to label the tree for felling. The views may be zoomed, the colours of tree species may be changed from their defaults and trees may also be planted using the same graphics facility. Features which may be improved upon are minor, but include the size of dialog boxes, the ordering of window appearance and the height of trees relative to their horizontal position.

If purchased, the code for generating the graphics would be required in order that it could be properly integrated to the modelling framework. Prices would be negociated for this. In addition to the financial cost, however, some considerable time would be required to implement the graphics within the code. This should be compared to the time needed for similar graphical software to be written in-house, and consequent reduced inconvenience of maintenance of the graphics.

11 Other models reviewed

BWIN 2.5 (Nagel, 1997) is a regression-based growth model for forest stands in Northwest Germany. Whilst not appearing particularly relevant to this project, they have some features in common:

- It requires more data than is collected in the forest, so a data generator is used to fill in the gaps, for crown height and crown diameter as well as tree diameter, age, height and position. It is likely that some data generation will be necessary for process-based growth modelling in Indonesia, and BWIN could be referred to.
- Thinning: there are 3 thinning methods available: by defining a certain diameter above which all trees will be thinned; by selecting trees using sensitive graphics; and you can perform an "A-value thinning". In the last case crop trees are favoured. You can select crop trees interactively on the screen. These will also have to be considered for selection of logging options to implement in the Indonesian project, so again, BWIN may be referred to.

SmartForest (Orland, 1997) is an interactive forest modelling and data visualisation tool. It is clearly designed for North American temperate forests, although some of their ideas would also be useful in an Indonesian context. In particular, the idea of interactive visualisation would be useful for selective logging. SmartForest connects the graphic with an underlying database, allowing dialog access to the database following selection of a tree (or stand) from the graphic. For selective logging, certain data are particularly relevant, and these could be connected to the graphic in a similar manner. It is not proposed that any direct use be made of SmartForest.

12 Programming languages

There are a few requirements of programming languages to be used as front-ends in this project:

1. They should be "visual", for the windows programming.
2. They should be able to handle calls to routines written in C (from old SYMFOR routines) and Fortran (from various segments of Hybrid code).
3. They should be able to handle ODBC and ActiveX, for the clearing-house database input.
4. They should be fast, modern, and have a friendly development environment.
5. They should be widely used and well established, so that it is not possible they will become obsolete in the next 10 years.

There are many visual programming languages: included amongst those considered were Visual J++ (Microsoft's Java), Delphi, Visual C++, Visual Basic and Tcl/Tk. Tcl/Tk is a fully portable scripting language, which, while having its advantages, is not ideal for handling data or calling routines and it does not support ODBC/ActiveX. Visual Basic is an interpreted language (which will run slower than a well-written, compiled code), although it is simple to understand and the current SYMFOR is written using it. Visual J++ is also an interpreted language, and, since there is no proposal to use specifically Java objects in this project, it's employment appears unnecessary. Delphi satisfies most of the requirements above, although it was difficult to ascertain whether calls to C and Fortran routines were supported. C++ certainly satisfies all of the above, and has some additional advantages: it is supported by Microsoft's Fortran Powerstation; the C code

from SYMFOR will compile in C++, reducing the compilation process by one stage; it is recognised as the world standard for powerful object-oriented programming.

It is the author's view that while Delphi and Visual J++ (and probably several more languages) would be at least satisfactory for this project, since one has to be chosen, Visual C++ has the most advantages relevant to this project.

13 Data I/O

The PSP data collected from the clearing house in Indonesia are in the form of Microsoft FoxPro databases. These data should be input to a framework model without conversion of the format, in order to increase the speed and simplify the use of the framework. It is also necessary to allow options for the output format of the data, as different users will have differing favourite post-processing analysis packages. Output formats that have already been suggested are FoxPro, Excel, SAS and as text files.

Microsoft Visual C++ v5.0 supports ODBC (Open Database Connectivity) API (Application Programming Interface) function calls. It also has a FoxPro ODBC driver, which allows SQL (Structured Query Language) commands to be executed on the FoxPro database by C++ ODBC API calls. This means that the C++ code can directly access the data in the FoxPro database. The MFC (Microsoft Foundation Class) library contains classes specifically for handling input from ODBC objects, such as databases and record-sets, which simplifies the programming necessary.

Output to ODBC objects must be carried out through direct calls to the ODBC API functions. Microsoft Visual C++ v5.0 also has the Excel ODBC driver, and a text file ODBC driver, which could be installed if necessary. C++ ODBC API function calls are not driver-dependent, which means that FoxPro, Excel or text files could be written with the same code, by changing a switch. This greatly simplifies output format possibilities of the framework.

14 Summary

Three realistic proposals have been established as potential development strategies for the GYM project:

1. A modified SYMFOR-style approach that sets the module choices to be post-compilation and utilises a Hybrid-derived tree-object for growth.
2. A modified Hybrid-style approach, adding management options and a SYMFOR-like front end onto the existing version of Hybrid.
3. AME, with the construction of graphically visualised models and sub-models, and TCL helper functions enabling the modeller to quickly put together a working growth and yield model.

In addition to those mentioned above, several other models were reviewed. Many of these have particular design features which would be of advantage to a framework output from this project. These have been noted for future reference. They do not demand immediate attention at this stage as they are applicable to all possible development strategies that this project could adopt.

15 Recommendations of the author

This section is to be used as a basis for discussion leading to an agreement on the strategy for the development of the GYM project. The opinions given are based partly on the restrictions and criteria given in sections 2 and 3, but also on the ease of implementation of the various options and the perceived likelihood of success or occurrence of problems during development. Many of the issues which should be used to compare the strategies have already been stated in the reviews of the various codes, and are not repeated here.

In my view, the two most beneficial features that the considered frameworks could have are:

- run-time module choice-taking, and
- having an working version ready early.

The former allows a simple software design to be used, aiding maintenance of the code and the development process. It reduces the number and complexity of input files that may be required. It allows batch processing of commands for extensive multiple runs, and allows non-modellers to select different modules without feeling like they are making new models.
The latter guarantees that the project will succeed at some level at least. It allows users of the software to produce results using something other than the current SYMFOR, for example, in the evaluation of criteria and indicators. and allows users to become accustomed to the software and report problems during development. It also encourages the development of the project in the most productive direction at all times.

Accounting for these features, the most advantageous strategy to adopt would be number 1 (see the summary, section 14). Its primary advantage, over and above those of the other proposals, is that it is based on the current state of SYMFOR. This is an advantage because SYMFOR already satisfies many of the requirements on the new framework. Most of the many changes which are proposed are incremental: for each of these there is already a working version in SYMFOR which could be used with minor modifications. For example, the Visual Basic model manager need not be replaced with a C++ version; the functionality of the current version is similar to that which will be required for the new framework, so a working Visual Basic model manager would be ready in far less time than a C++ one. This ensures that there will be a positive outcome from the project even if it does not reach the state proposed in the initial design. It also ensures that a working version (albeit not the final version) will be available long before the end of the project, allowing the evaluation of criteria and indicators to proceed. A work-plan for this strategy would be based on this idea, so that the essential re-coding would be completed

as soon as possible and other, more aesthetic, changes would be made after a working version is ready. This system would allow all of the features mentioned as desirable in section 3, and in particular it is independent from software that is maintained outside the responsibility of this project. This means that it does not depend on the continuing support of other packages for its own sustainability.

Strategy 2 is also based on existing code (Hybrid) which would speed development. Hybrid was not developed for use as a forest management tool, however, and large sections of code (the model manager and the management options) would need to be written before a working version could be achieved. FORMIX (Huth, 1997) (see section 9), which has already been under development for 6 years by the Kassel group in Germany, has largely the same targets and methodology as described in strategy 2. Although there may be issues concerning intellectual rights and collaboration with the Kassel group, the sensible way to achieve the final results proposed in strategy 2 would be by collaboration with Kassel, and would probably largely ignore the current version of Hybrid. Although this may be impractical, production of a similar system using Hybrid would not represent an advance of the subject, and would not provide the tests of the methodology for comparison which would be possible for the other strategies. To modify a framework (e.g., add an optional routine for a certain process), a modeller would have to program in two languages, and would have to modify the existing Hybrid code. This could cause problems of multiple copies of the source code emerging, if the Hybrid code is also under continual development at ITE. A run-time module choice would be possible, although I think the real question here is whether there would realistically be many options to choose from: in my opinion, the current Hybrid routines would not lend themselves to simple substitution by routines written by other modellers without extensive documentation of the code.

The AME proposal (strategy 3) would require a separate model for each possible combination of routines (or sub-models). The user could choose this from a list of existing models, but if the required one does not exist it would need to be made using AME model visualisation. The lack of existence of code is even more relevant for strategy 3 where neither the models (in the form of the AME visualisation) nor the handlers (TCL helpers) or model manager yet exist. These would have to be written simultaneously, and then all possible models (combinations of sub-models) would have to be produced before realistic use of the system was possible. There would, as far as I can see, be little scope for having an early working version which could be used while development continued, and no scope for having a run-time module choice.

References

Friend, A.D., Stevens, A.K., Knox, R.G. & Cannell, M.G.R. (1997) A process-based, terrestrial biosphere model of ecosystem dynamics (Hybrid v3.0). *Ecological Modelling*, 95, 249-287.

Huth, A. (1997) Rain Forest Simulation Model, FORMIX. *WWW page*: http://dino.wiz.uni-kassel.de/model_db/mdb/formix.html,

Lawson, G.J., Cannell, M.G.R., Mobbs, D.C., Crout, N.M.J., Muetzelfeldt, R., Wallace, J.S., Gregory, P.J., Thomas, T.H., Jagtap, S., Ludlow, A.R., Cobbina, J., Arah, J., MacDonald, K.J., Taylor, J., Sharp, L., Wiggins, G., Flynn, L., Livesley, S., Willis, R.W. & Friend, A.D. (1996) Agroforestry modelling and Research Coordination, Annual Report to the ODA, chapter 11. 124

Leersnijder, P. (1997) RAFOM, a product of Forest Graphics. *WWW page*: http://www.wirehub.nl/~teleoffice/business/fg,

Muetzelfeldt, R. & Young, A. (1996) Sustainable Yield Modelling for Tropical Forests (SYMFOR user manual), Report to the ODA. RES/MOD/96/2,

Nagel, J. (1997) BWIN 2.5, A forest growth model for Northwest Germany. *WWW page*: http://merlin.uni-forst.gwdg.de/nfvabw02.htm,

Ong, R.C. & Kleine, M. (1995) DIPSIM: A dipterocarp forest growth simulation model for Sabah. *FRC Forest Research Papers*, 2,

Orland, B. (1997) SmartForest: An interactive forest data modelling and visualisation tool. *WWW pages*: http://imlab9.landarch.uiuc.edu/smartforest/index.html,

Rombouts, J. (1997) Data Base Structure of the Growth and Yield Clearing House.

van Gardingen, P.R., Friend, A.D., Heuch, J. & Wirodidjojo, I.S. (1997) Growth and Yield Modelling Project Memorandum. *WWW page*: http://meranti.ierm.ed.ac.uk/frp/,

Appendix A: Definitions as used in this document

Model: A system of relationships designed to enable a user to simulate the behaviour of some system (e.g., Hybrid). The relationships in a model may or may not be parameterised.

Modelling framework: A system of relationships designed to enable a user to simulate the behaviour of some system, including a facility for implementation of alternative relationships with which the user can describe a particular process or feature of the system (e.g., SYMFOR). It may be possible to develop a model within a modelling framework by a choice of relationships.

Modelling environment: Software which simplifies, in some sense, the process of the implementation of a model (e.g., AME). It may be possible to develop a model or a modelling framework within a modelling environment by the implementation and subsequent choice of relationships.

Empirical, or regression-based: In modelling a process, where a measurable quantity is evaluated directly from the measurable factors affecting it (e.g., modelling growth using the quantity of light available with some appropriately parameterised coefficients).

Process-based: In modelling a process, where a measurable quantity is evaluated by a series of relationships, stemming from measurable factors, but relying on an understanding of the underlying functionality of the process at some level. (e.g., modelling growth using outputs of the model of the process of photosynthesis and appropriate parameters, which in turn relies on the quantity of light available and other parameters). For this reason, process-based models often use more parameters, and consequently require more data in order to evaluate them unambiguously, than empirical models.

Modular: Software constructed in such a way that each process or task performed is completely separable from the rest of the code and therefore operates using input and output data, without sharing data with other sections of the code. Routines using externally defined data are not fully modular according to the definition used in this document. This definition gets a bit shaky when discussing externally defined data structures, as is common in C or C++.

Parameterise (or calibrate): Evaluate the parameters in a relationship. Parameters are the coefficients which are necessary in order to render the relationship valid for the circumstance in which it is intended to be used. The parameters are distinct from the relationship variables which may be either data input to the model, or data calculated by the model, although the parameterisation may place validity limits on the value of the relationship variables. The parameters are also distinct from universal constants, (e.g., π). **Note:** this definition of "parameters" differs considerably from that used elsewhere in the subject of ecological modelling, which is: that set of quantities which are kept constant throughout the execution of the model. "Parameterisation" is still defined as the evaluation of the parameters.