UNIFIED MODEL DOCUMENTATION PAPER NO. 2

Change control management of the Unified Model System

by

R. Rawlins

VERSION 1.5

2/12/98

MODEL VERSION 4.5

Numerical Weather Prediction
Meteorological Office
London Road
BRACKNELL
Berkshire
RG12 2SZ
United Kingdom

Modification Record		
Document version	Author	Description
Draft 1.0	R.Rawlins	First version. 07/06/94
1.1	R.Rawlins	Add paras 6 and 7 on testing procedures and some minor updates. 27/06/95
1.2	R.Rawlins	Minor changes accompanying mod edit facility and responsibilities. 08/08/96
1.3 (Vn4.3 of UM)	R.Rawlins	Minor updates following migration to the T3E. 15/07/97
1.4 (Vn4.4 of UM)	R. Rawlins	Fuller inventory for vn4.4. Change to Word document. 26/01/98
1.5 (Vn4.5 of UM)	R.Rawlins	Changes following release of vn4.5 applicable for the next version. 02/12/98

Contents

1. Introduction

The Unified Model (UM) system consists of a collection of integrated software files and the documentation needed to describe them. There is a need to preserve the integrity of code for a large community of users but also allow development of new facilities and correction of errors. This paper describes procedures that have evolved to manage those changes. It is prepared at the end of a cycle of development, ie a UM version - and is labelled at that version - but is applicable to management of succeeding versions.

UM users have requirements for access to code that is stable over a period of months and years to allow development and testing of scientific modifications. Basic properties of the UM system are: the large number of interrelated facilities, the number of parallel developments of new code and the scope of different types of model configuration. Extensive sharing of code between different model and the cost of performing tests in all likely permutations of model types has led to a change control mechanism based on successive versions. Here testing of the entire system is performed centrally on selected model configurations using the entire set of changes submitted to the system. Once testing has been successfully completed a new version is released and all components of that version are locked. A more severe constraint for some categories of model user is a guarantee of reproducible results at the bit level - UM Doc Paper 4 describes methods adopted, which involve maintaining separate releases of scientific modules.

2. UM system inventory

2.1 Application software

2.1.1 UM files and source code

These provide: selection of source code for a model configuration, creation of an executable file and control of a model run. Master copies are held on the CRAY T3E under directory /u/um1 (=system environment variable $UMDIR). Each version is stored separately under a specific version no. within a sub-directory /u/um1/vnxxx (where xxx=version number, e.g. 4.5). A UM version (vn) is defined by the complete set of components (plus UMUI files). Principal constituents are held in sub-directories below:

/source: source code for Fortran and C are held in nupdate program library umpl; source code for unix scripts in program library umsl; source code for generating model specific ancillary files from primary data in program library ancilpl;

/scripts: unix scripts;

/exec: internal UM executable files;

/obj: pre-compiled object code;

/ctldata: system data files;

/ancil: primary and model specific meteorological ancillary data built from primary data (see UM Doc Papers 70 to 73).
/utils: general purpose utility programs (see UM Doc Paper F5).

2.1.2 UM user interface (UMUI)

This is a graphical user interface providing the set up capability for model execution by creating script and control files on a workstation platform. Master code and files are located below directory ~umui on a HP workstation (currently hc0300).

2.1.3 Workstation utilities

Non critical programs exist to facilitate use and management of the UM. These are held locally on workstations in /usr/local/bin. Examples include:
modedit: a utility for lodging future changes to source code.
umsubmit: an alternative method of submitting UM jobs.

2.2 Third party software

GC/GCOM libraries for generalised message passing communications. Held under directory /u/um1/gcom on the T3E.
OASIS libraries for coupling non-UM models. Not maintained centrally.
TCL/TK toolkit (public domain package). Held below directory ~umui on a HP workstation (hc0300).

2.3 Documentation

Documentation is available on-line via the Met. Office intranet (MetWeb), with top level entry at http://www-nwp/umdoc/hello.html and as UM Doc papers, with printed copies available from the UM Librarian. Documents are labelled by the version that they are valid for, with the exception of this paper (002), which refers to development of the following UM version. Documents may be released at a time subsequent to the release of software for a version.

3. Source code maintenance

Source code is managed using nupdate (originally a CRAY proprietary package, but now in the public domain), which allows modification sets (mod-sets) to be developed in parallel by different users with reference to a particular version. It has the advantage that separate developers can isolate their work relative to stable libraries. Difficulties arise when changes are not orthogonal and clashes of modification sets require individual treatment. This makes it particularly important that advance notice of work spanning commonly used modules is given.

3.1 Sequence of change control

The sequence of change control steps is as follows:

(i) Notification of expected changes that impact the model system are ratified through the UM System Manager, and through WGDUM. These are added to the list of outstanding tasks (documented in the Met. Office Web). The selection of major control changes for a new UM version is project managed within the forum of IEG (Infrastructure Experts Group).

(ii) An explanatory change header is placed on-line in the UM changes directory on the T3E - at vn4.5 this is /u/um1/vn4.5/mods/source - by the change author, based on member @CHGFHED in the same directory as a template. This is achieved automatically by use of the *modedit* facility on workstations. The change header also contains a brief design specification, which should be completed before testing takes place and be available for assurance by a reviewer.

(iii) The change mod-set is developed and tested, and added to the change header already in place on the T3E file. The initial creation of a mod-set in (ii), and subsequent edits to it, must only be performed via the *modedit* facility, which places each mod-set under RCS control. This preserves incremental edits of each mod-set, provides a record of development and prevents authors making conflicting simultaneous changes. Modedit is a program developed in-house and is invoked on HP workstations either as a script or as file type function selected by mouse click. Entering *modedit -e myfile.mod* allows entry of text to describe each incremental edit - more information can be obtained by entering *man modedit*;

(iv) The author fills in a (hard-copy) change form and sends to the reviewer of the change;

(v) Once the author has responded to reviewer requests and the reviewer is satisfied with the code change, the change form is signed and submitted to the UM librarian;

(vi) The UM librarian will check mod-sets using a set of tests for known errors and deficiencies (line length exceeding 72 columns, missing comments in headers, use of *COPY, and any others that compromise the integrity of nupdate processing). This check can be instigated at any stage.

(vii) [Introduced at vn3.4] Periodically, as modifications are received, the librarian will build an interim source library based on the previous version plus the sum of all submitted changes. If clashes with earlier code changes occur, it is the responsibility of the author of the later change to resolve the problem;

(viii) Once the deadline for changes is reached, new source libraries are built and other components are updated to form a provisional UM version;

(ix) A standard set of tests are performed covering a variety of model configurations and model changes are amended only if errors are encountered;

(x) Provisional libraries are re-built as necessary until all errors uncovered by central testing are resolved;

(xi) A new version is deemed to be released, no further changes are permitted within constituent files; this information is broadcast and a timetable given for the next version.

This procedure is designed to encourage sufficient checking of modifications to minimise the introduction of errors while maintaining the choice of parallel development. The steps introduced at vn3.4 require more advance notice of work to be undertaken and spreads the load of reducing code

changes over more authors and over a longer period. Technical methods employed to build new executable files and compiled object code are described in UM Doc Paper Z5.

Note that all code modifications are relative to the previous version so that they can work independently (if not linked explicitly to another mod-set). For step (vi) a record is kept of the order in which completed mod-sets are received by the librarian; successive interim builds apply all mod-sets in this order and are only used for resolving clashes. However authors can access the interim build library for this purpose themselves. In the event of a clash:

-The librarian should be informed and the mod-set copied to the central library;

-The position of the mod-set in the record is not altered and the interim library is rebuilt and checked again for clashes which should be resolved by the author of the changed mod-set.

3.2 Timetable of change control

Summary of normal life cycle for versions:

Label	+ Months	Deadline
RELEASE	0	Release of current version and dates set for next version
HEADER	2	Mod-set header in place on-line giving details of expected change
REVIEWER	4	Mod-sets completed, last date for changes to be with reviewers
FINAL	4.5	All reviewed mod-sets received, changes only allowed to resolve clashes, new version of UMUI ready
BUILD	5	First build of object code and executable files, start tests of entire new version
RELEASE	7	Release of new version

The timetable in months is only approximate and will be specified for each release. The dates given are appropriate for about 2 versions per year, with testing likely to occupy 1-2 months but being potentially open-ended. Note that *__REVIEWER*__ is the __*last*__ date for changes to be with reviewers - they would generally expect much greater notice and can negotiate earlier dates. Plans will be made on the basis of change request headers issued by *__HEADER__*; any late changes will need to be accepted by the system manager, normally only error corrections would qualify.

Between *__FINAL*__ and __*BUILD*__ dates, no new mods will be accepted - this period will only be used for making corrections to resolve clashes. Only reviewed changes (by *__FINAL*__ date) will be included into the system.

4. Nupdate mod-set identifier

Up to vn3.3 nupdate mod-set identifiers (*IDENT) were of the form NN071293, giving the programmer identifier and date. From vn3.4 a new label was adopted with a structure @NNSL### where:

@ is a guide to the area of code being changed -@

=A atmosphere sub-model
=O ocean sub-model
=S slab sub-model
=C coupling between sub-models
=G general control/system changes not linked to a specific sub-model
=U utilities (executable files)
=V variational assimilation and tangent linear model

ie if the modification is predominantly to code lying under a *DEF ATMOS switch, then @=A. It is recognised that not all changes can be unambiguously in a single category - the intention is to allow users to browse related modifications more easily.

NN is a 2-character programmer identifier

S=0-9,A-Z is a 1-character mod-set label to distinguish between mod-sets generated by the same author

L is a language identifier:

=F Fortran code
=C 'C' code
=U Unix scripts

is the version number

ie 405 for version 4.5

This information is repeated in the template header on the T3E in directory /u/um1/vn4.5/mods/source. For example a Fortran modification owned by programmer az, affecting atmosphere code could be labelled AAZ1F405. Note that date information can be obtained by inspection of comments in headers within each subroutine module; however the version no. is the principal marker for looking at previous changes.

Mod-set files on the T3E have the same name as their *IDENT identifier, but in lower case, i.e. aaz1f405. The equivalent files on HP workstations are suffixed .mod, ie aaz1f404.mod, so that the modedit facility can be invoked.

5. Ownership/reviewing

It is the responsibility of the reviewer to ensure that a code change fulfils the requirement of the change, has minimal implications to other users, is adequately documented and conforms to UM programming standards (UM Doc Paper 3). The reviewer can request evidence of completed tests to establish that the requirement has been met.

The UM project originally attempted to allocate an owner and reviewer for each individual sub-component of code. Experience has shown that the structure of the model has grown and diversified faster than the capability of the Divisions to provide and train staff, and faster than a fully integrated documentation system to cope with. In particular there is not the flexibility or number of resources available to provide dedicated reviewers for every area of model code. There is a need to spread reviewing tasks for control changes more widely between groups, so that these are checked more thoroughly for impact in unrelated areas. However review tasks within a discrete module of the model, such as the dynamics or ocean code, can usually be handled within the group itself. Ownership issues can become unclear when responsibility is split down to small sub-sections of the model because of interface implications and gaps between sub-sections. It is more easily resolved when all sub-components lie within a relatively large module under the same owner, typically a group leader who has some flexibility in assigning local tasks within his own group.

In response to the previous points, the following guidelines have been adopted:
(i) The component list has been reduced to a smaller set of large modules (where some overlap may occur); each module owner may sub-divide responsibilities locally within his group; a current de facto list of components with owners is held by the UM System Manager - see http://www-nwp/~frrr/UMversions/UMOwners4.6.html Ownership of physics modules varies with the area of active development - ownership will be established on a version by version basis, early in the cycle of development of the code.
(ii) In the first instance it would be expected that the person who requested a change would be approached to be the reviewer.
(iii) A reviewer would be obtained from a pool of reviewers named by (NWP), (CR), (OA) representatives at IEG.
(iv) Owners of code are not expected to be the sole contributors to changes of their code but must always be informed.
(v) Final responsibility for code changes lies with the owner of the code at that version- this is mainly applicable for the owners of science sections.

6. Test procedure after changes deadline

The librarian resolves any remaining modification clashes as described in para 3 and prepares a simple concatenation of all modsets that apply to the model. This is not mandatory but can be used by testers to obtain a first look at compile problems. If problems are found they should be communicated to the author of the offending modset(s) and corrected. In all cases of amendment after the changes

deadline, corrections should be made to the original mod-set, with a dated nupdate comment at the top describing the change. It is preferable to add extra code to an existing relevant mod-set than to generate a new mod-set after the deadline. At this stage changed modsets are only introduced into a central build directory under the librarian's control, so that it is known which modsets have been newly modified.

The scripts' owner builds a new version on the T3E from the previous code plus modifications, i.e. generates new source code libraries, new scripts and executables and object code; UM Doc Paper Z5 lists detailed procedures. Problems identified during the build should be corrected as before until errors in object code compilation have been removed. The new version of the umui will generally be available before this time, but should be completed by build time to point to the new libraries. It will then be ready for end to end testing of a model from initial set-up in the umui to running on the T3E.

Different categories of user will then start testing specific configurations, make commented corrections to modsets (which may be referred to original authors) and communicate with the librarian, who will broadcast details of findings to other testers. The version will be rebuilt at intervals when a batch of corrections have been accumulated and testing resumed until tests have been found to be satisfactory. At this point all libraries and files are frozen, and an announcement is made to all users stating the release of a new version.

7. Criteria for release

Test criteria only refer to model runs on the Cray T3E. The version will be released when testing has been successfully completed for the standard configurations:

7.1 Standard configurations

7.1.1 Experiments (Mandatory)

Global operational
Mesoscale operational - UK
Mesoscale operational - defence
HADCM3
HADCM4
FOAM ocean model

7.1.2 Experiments (Desirable)

Non-MPP global and Mesoscale operational
HADAM3
Global at lower resolution (288*217)

Operational stratospheric model
Ocean models - various other
Portable models on workstations

Other experiment configurations, e.g. HADAM2b, SLAB, WAVE may be corrected by users outside the project during the testing stage, but will not form part of the acceptance criteria.

7.3 Other UM functionality

UMUI
Scripts and compile system
File utilities
Output file utilities

7.2 Test criteria

Each mandatory experiment configuration should:

a) be capable of being set up through the UMUI, without subsequent editing of control files generated by the UMUI ;
b) be capable of upgrading model dumps from a previous version and be upwardly compatible from the previous version via the reconfiguration;
c) compile and run to normal completion, including generation of standard output and archiving;
d) provide continuation runs;
e) demonstrate that any cpu timing and memory changes are understood and acceptable to climate and operational users .
f) Forecast model dumps created with the previous version code , plus an accepted subset of modsets), should be bit-reproducible with those created with the new version , for tests with the same compiler at both versions. *Unconditional departures from bit reproducible answers at the new version are permitted only if:*

(i) Reasons for the departure are understood and quantified;
and, (ii) any impact on operational scores is negligible or signed off by WGOS;
and, (iii) (CR) representatives accept changes for the HADCM4 configuration;
or, (iv) it is accepted that the released version will not be bit reproducible [, such as could be required by the adoption of a new compiler].

g) Operational configurations should run successfully across Y2000.

In addition, it is desirable that:

h) All configurations be entirely recompiled and run with run-time bounds checks;
i) Low resolution configurations should be entirely re-compiled in non-MPP mode to check normal running .
j) New functionality should be tested by the code developer, from checking UMUI panel entry to final product.

7.3 Bit reproducibility

The criteria to be adopted for central testing towards releasing a new version depend upon the classification of the new version, which will be notified well in advance of its development and may be:

i) an interim version, required to split a large set of structural changes into more manageable components, which is part of the development process but is not supported as a general release for users;
ii) a non-bit reproducible change, where data values will not necessarily compare at the bit level with the same experiment at an earlier version.
iii) a bit reproducible version change, in which a specified set of configurations generate the same data values within forecast dumps in comparison with the previous version. All version changes will fall into this class if it can be achieved;

Problems arising from the additional set of configurations, and any output problems notified by users, will be corrected as time affords but need not delay release of a new version. In general a version will be released once (CR),(NWP) representatives are satisfied that tests are sufficiently complete. The final decision on version release will be made by the UM system manager. It is not mandatory that all corrections are assimilated into the released system, provided that the source of the problem has been identified and users can circumvent it at the new release, usually through inclusion of local source code modifications.

7.3.1 Interim version

The minimum requirement is that the base set of configurations should compile, run to completion, generate and receive boundary conditions as appropriate, allow continuation runs and produce data output. Processing of umui set-up windows should generate control files without the need to make local amendments. Overall cpu costs and memory requirements will be checked to ensure that no unplanned increase is introduced.

It will be usual practice to change compiler directives to allow array bounds checking and subroutine argument checking at run-time. New problems arising from these tests should be explored.

Forward compatibility is a requirement, i.e. it will be ensured that dump or boundary files generated from earlier model versions can be used at the new release, so that the new model can be applied to old case studies. This will be achieved automatically where possible, but conversion through a utility is also permissible. Backward compatibility will also be checked, to investigate whether users running old experiments starting from new version data files will encounter difficulties. There is no guarantee that backward compatibility will be maintained and it is always advised that users move up to the latest version.

Models should be capable of achieving bit reproducible results across multiple new runs, with or without continuation runs.

All models should be capable of running on different configurations of processors (PEs) in MPP mode, and would normally be expected to give bit-reproducible results - any departure would need to be ratified by the UM System Manager before inclusion. For vn4.3 on, assimilation in the atmosphere model is known to lose bit reproducibility for different numbers of PEs. In addition all models should be capable of running on a non-MPP platform, i.e. providing that memory requirements permit - the functionality of model code may require testing on a large memory PE or at a lower resolution to prove this.

7.3.2 Bit reproducible version change

The criteria in 7.3.1 need to be satisfied plus the requirement that prognostic field data within forecast dumps for a standard run of the climate configuration and of the atmosphere-ocean configuration should bit compare with the previous release. It is necessary that this should be achieved in a mode (i.e. not necessarily automatically) so that it can be demonstrated that numerical differences have not been introduced by control changes, i.e. there should not be an unconditional loss of bit comparability. Bit reproducibility in operational configurations will also be tested to check for the introduction of errors. However it is not a mandatory requirement that operational configurations should bit compare across versions.

7.3.3 Non-bit reproducible version change

The criteria in 7.3.1 need to be satisfied plus the requirement that models forecasts should be essentially unchanged by the move to a new version, i.e. science and control changes should be independent. In this case it is necessary to perform further tests that any differences that arise between old and new versions are minor arithmetic differences that can change the detail of a forecast but maintain the same climatological and average synoptic response. (CR) representatives need to be satisfied that the climatology is effectively the same for old and new releases. Operationally (NWP) representatives need to be satisfied that verification scores from old and new releases are closely similar. It is difficult to quote quantitative values for these criteria - the requirement is that differences between new and old versions is of the same order as differences obtained from very minor perturbations of the model set up, such as by choosing a different compiler optimisation option, or a different frequency of dumping.

8. Code changes after release

It is expected that modifications after release will be confined to the user level, i.e. a user will make local changes to source code, scripts and other files through the mechanism of adding modification sets to their own UM job.

8.1 Exceptions

Once a version is released, source code in system libraries is unconditionally frozen. Scripts, system data files and executable files are normally fixed after release: the only exception is when a computer system change or gross error affecting a majority of users requires a mid version update, which must then be communicated to all users. If any files are changed for this reason, a local README file in the same directory should describe changes and the method of building modified files.

Corrections to UMUI files may continue after release, providing there is no impact on users, in which case a 'mini-release of the UMUI may be required to protect established experiments' set-up.

8.2 Errors after release

When errors in the latest release are discovered (e.g. at vn4.5) and explained, it is convenient to place a nupdate correction mod-set in the $UMDIR/vn4.6/mods/source directory. These mod-sets fall into 1 of 2 categories:
i) a fix for the current release which is not suitable for building permanently into future versions, ie the correction may have a limited range of applicability or may be superseded by other changes;
ii) a correction which is complete and meets UM code standards - this will have the status of any mod-set forwarded for inclusion at the next build.

The mod-sets will be lodged via modedit, and the two types will be distinguished by a question in the mod-set header. Any changes will be lodged and recorded via the RCS mechanism within modedit. A summary of differences to all files in the UMDIR/vn4.6/mods/source directory will be automatically displayed on a daily basis in the UM newsgroup.

Once a correction has been found the information should be broadcast via the UM newsgroup. In addition a permanent record should be made by logging the problem in the UM Problems Database.

9. Announcements

General announcements concerning release dates, model errors and items of interest are made in the UM newsgroup ukmo.met.um. Errors specific to the UMUI are broadcast in newsgroup ukmo.met.umui-bugs. After the final deadline for modifications has been reached, communications concerning findings for the build and tesing stages of a new version are sent to the ukmo.met.um.test newsgroup.

Acronyms

CR	Climate Research Division
FOAM	Forecast Ocean Assimilation Model

HP	Hewlett-Packard
MPP	Massively Parallel Processing
NWP	Numerical Weather Prediction Division
PE	Processing Element
UM	Unified Model
IEG	Infrastucture Experts Group
UMUI	Unified Model User Interface
WGOS	Working Group for Operational Systems
Y2000	Year 2000