Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

APM

Software Management

Rob van der Linden

Abstract

For APM to achieve ISO9000/BS5750 accreditation, there is a need for better management of its software. This includes extensive version control for all software, developed in a multi-project environment with controlled reuse of code.

The procedures in this document are designed to allow complete tracability of code changes throughout the software life-cycle. A source code management system is to be introduced which can (eventually) cater for the quality procedures introduced in this document.

The document should be read in the context of APM's current projects and future activities in Object Lab. Object Lab will add significantly to the body of software which will need to be produced and maintained by APM staff. It will also increase the opportunity for code reuse and consequent productivity improvements.

APM.1697.00.01 **Draft** 17th January 1996

Project Management (confidential to ANSA consortium for 2 years)

Distribution:
Supersedes:
Superseded by:

Software Management

Software Management

Rob van der Linden

APM.1697.00.01

17th January 1996

The material in this Report has been developed as part of the ANSA Architecture for Open Distributed Systems. ANSA is a collaborative initiative, managed by Architecture Projects Management Limited on behalf of the companies sponsoring the ANSA Workprogramme.

The ANSA initiative is open to all companies and organisations. Further information on the ANSA Workprogramme, the material in this report, and on other reports can be obtained from the address below.

The authors acknowledge the help and assistance of their colleagues, in sponsoring companies and the ANSA team in Cambridge in the preparation of this report.

Architecture Projects Management Limited

Poseidon House
Castle Park
CAMBRIDGE
CB3 0RD
United Kingdom

TELEPHONE UK	(01223) 515010
INTERNATIONAL	+44 1223 515010
FAX	+44 1223 359779
E-MAIL	apm@ansa.co.uk

Contents

1 Source code management

1.1 Introduction and scope

For APM to achieve ISO9000/BS5750 accreditation, there is a need for better management of its software. This includes extensive version control for all software, developed in a multi-project environment with controlled reuse of code.

A source code management system is to be introduced which can cater for the quality procedures introduced in this document. These procedures are designed to allow complete tracability of code changes throughout the software life-cycle.

The document should be read in the context of APM's current projects and future activities in Object Lab. Object Lab will add significantly to the body of software which will need to be produced and maintained by APM staff. It will also increase the opportunity for code reuse and consequent productivity improvements.

1.1.1 Why this document?

This document is intended as the definition of the *software control process* which will need to be supported by any software source control (or configuration control) system. As such it can be used as a requirements document for such a system.

The aim is to define the processes very well and rely on automation of such processes as much as possible. This will reduce the effort involved in following quality procedures and eliminate errors.

1.1.2 Who should read this document?

This document is aimed at:

1. those who are involved in setting up quality systems at APM
2. those who will be involved in the selection and purchase of the software source control (or configuration control) system
3. those who will be using the software source control (or configuration control) system

1.1.3 Other documents

We have borrowed from the procedures which were set up for the document system [APM.1026] and adapted these where necessary.

There will be other documents, such as style guides for code writing, test guidance, etc.

2 Software version control

2.1 Definitions

2.1.1 Module

1. A module is the basic unit of version control
2. A module is defined by its interface (IDL) and a set of attributes as follows:
   - name
   - number
   - issue
   - version
   - part of release?
   - release ID
   - status
   - project
   - author
   - date coded
   - date tested
   - change history record
     - problem reports, implemented and dates
     - change requests, implemented and dates
   - dependencies (modules called)
   - full description (or reference to full description)

2.1.2 Project

1. A project is a set of related activities which aim to produce one or more (related) modules.
2. Modules created within a project are owned by that project.
3. Ownership of modules may imply restricted access to modules for reasons of IPR for instance.

Note: Project organisation: no APM document yet, but in preparation.

2.1.3 Release

1. A release is a set of modules which are offered to organisations outside APM
2. A release is clearly identified by a number

3. A release is reproducible (the revision of all modules that make up a release are to be kept)

2.1.4 Library

1. The Library holds a set of modules which are considered to be generally re-usable
2. A module in the Library is seen as an “internal *release*”.

Note: Do we need library releases? In that case a library is an internal release

2.2 Project

2.2.1 Project life cycle

1. A project has five phases (which may partially overlap)
   - — requirements phase
   - — functional design phase
   - — interface design phase
   - — coding phase
   - — testing phase
2. Before the requirements phase, the scope of the project is defined and documented in the Project Brief.
3. During the requirements phase the Project Brief is refined:
   - — the requirements are defined and the scope of the project is adjusted where necessary
   - — an approach to meeting the requirements is sketched
   - — resources and constraints are identified and documented
   - — the Project Brief is formally approved by the Technical Director and the Operations Director.
4. During the functional design phase the Functional Design Document is created:
   - — this FDD will become part of the manual to go with the project output
   - — a suitable technique (OMT diagrams and text) will be adopted for design
   - — the document is formally approved by the Technical Director
5. During the interface design phase:
   - — each of the interfaces to the modules defined in the FDD will be designed
   - — the use of IDL and of attributes for other characteristics is required
   - — interface definitions are reviewed and approved by the project team
   - — the final set of interfaces is formally approved by the Technical Director
6. During the module coding phase:

- each of the modules defined in the FDD and with interfaces defined in the interface design phase are implemented
- module testing is part of the module coding phase
- the review process includes code walk throughs and peer code review
- evidence of this review process must be presented to the Operations Director
- the modules progress according the module life cycle defined below
- the completed set of modules are formally approved by the Technical Director and the Operations Director

7. During the testing phase:
   - the set of modules will be assembled into a complete system or application
   - the result will be evaluated against the requirements set out in the requirements phase and the Project Brief
   - evaluation takes place by the customer or their appointed representative(s) (see Release life cycle below)
   - a release needs to be formally accepted by a customer before a project can end

2.2.2 Project results

1. A successful project results in
   - a fully documented release for use by our customers
   - project data for use by APM to help improve processes and planning of future projects
2. A project has a project archive which contains the documents associated with the project (for an example see [APM.1023]). In particular each of the phases of the project must be documented.
3. Note that documentation is an integral part of all activities in all phases
4. A project diary should be maintained in which relevant events, design changes, experiments etc. can be noted
5. The final project documentation should include a module parts list and a document list
6. Accounting information (timesheets, progress measurement and cost statements) form part of the project data.

2.3 Module

2.3.1 Module life cycle

1. A module is created in the context of a project
2. Modules can be in one of four states, reflected by the module status:
   - Draft
   - Approved
   - Withdrawn

3. The software librarian has the authority to change the status of a module.
4. The Librarian keeps
   — approved modules in the Library
   — draft and withdrawn modules outside the Library
5. The librarian acts on behalf of or after consultation with a review team
6. All modules start as Draft
7. Draft modules are subject to review, the outcome of which is a recommendation to:
   — leave the status as Draft
   — upgrade the staus to Approved
   — change the status to Withdrawn
8. Approved modules may be subject to review from time to time, resulting in a recommendation to:
   — leave the status as Approved
   — change the status to Withdrawn
   — update the module and change its status to Draft
9. Withdrawn modules remain withdrawn. It is permitted to re-use code from a withdrawn module in a new module if you think the new module can get through a review successfully.
10. A module may not be withdrawn unless all releases of which it is a part have Vbeen withdrawn

2.3.2 Module numbering

1. Software version control will manage all numbering.
2. A module number consists of three parts
   — number (unique in the context of APM)
   — issue (unique for each new issue to the outside world)
   — version (for internal version control)
3. All changes are managed by the version control system

2.3.3 Access to modules

1. Only approved modules may be reused
2. Modules may always be accessed by members of the project which provides the context for the module
3. Access to a module may be granted to members of (an)other project(s)

2.3.4 Module version control

1. modules can be checked in and out of the module file store.
   (i) A module may be checked out for:
   — reference (the same module cannot be checked in after this)
   — modification (all the rules about checking in apply).

2. A module which has been Withdrawn cannot be checked out for modification.
3. A module which has been checked in may not be checked in again.
4. A module which has been checked out for modification, may not be checked out for modification again.
5. A log is kept of what modules have been checked in and out, by whom and when. (This allows us to see how many modules are out, how long they have been out, etc.)
6. When a new module is started, it is checked out as a new module:
   (i) a module number is allocated (issue=0, version=1)
   (ii) a template module with the currently valid styles results in which all known information is filled in (project name, title, authors if known, date, module status)
7. When an existing module is checked out for modification:
   (i) the original module remains in the store
   (ii) the issue number of the copy being checked out remains unaltered
   (iii) the version number is incremented
   (iv) warnings about out of date formats are generated if appropriate
8. If an attempt is made to check out a module which has already been checked out, a message saying who has checked out the module, and when will result. The attempt will be logged.
9. When a module is checked in as a new version:
   (i) the version number is compared with the last checked out version
   (ii) the module database is updated
10. When a module is approved:
   (i) authorisation for the operation is sought
   (ii) the issue number is incremented
   (iii) the version number is removed (or set to 0)
   (iv) the module database is updated
11. When a module is withdrawn:
   (i) the issue and version numbers do not change
   (ii) the module database is updated
12. It is not possible to reverse the approval of a module other than by withdrawing it.
13. The action of checking out a module may be undone after the event (e.g. when it is decided that no changes are to be made, or no module is to be written after all):
   (i) an existing module reverts back to its previous VERSION number
   (ii) the number of a new module will NOT be recycled
14. All issues of a module need to be kept for reference, since queries can be expected from outside the project.

15. All versions after the current issue need to be kept for reference by the author and review team, and for audit.

2.4 The Library

1. The Library holds the current set of approved modules
2. The Library keeps approved modules for different projects separate

2.4.1 Adding and removing modules from the Library

1. Changes to the Library are upon recommendation by the Library Committee
2. Only approved modules are admitetd to the library
3. Library modules are subject to review from time to time, resulting in a recommendation to:
   - leave them in the Library
   - remove them from the Library

2.4.2 Responsibility for content

1. All modules in the Library are controlled by the Librarian
2. The Librarian is responsible for the consistency of the Library
3.

2.4.3 Access to the Library

1. Project members have free read access to the modules in the Library belonging to their project
2. For IPR reasons access control restrictions may exist for certain projects

2.5 Release

2.5.1 Release life cycle

1. A release is created in the context of a project
2. A release can be in one of four states:
   - Alpha
   - Beta
   - Released
   - Withdrawn
3. The project manager has the authority to change the status of a release
4. The project manager acts in consultation with the project steering committee, management board or similar entity
5. All releases start as Alpha
6. An Alpha release will be subject to a period of testing after which
   - it may be decided to keep it as Alpha

   - it may be upgraded to a Beta release
   - it may be Withdrawn
7. A Beta release will be subject to a period of testing after which
   - it may be downgraded to Alpha
   - it may be decided to keep it as Beta
   - it may be Released
   - it may be Withdrawn
8. A Release may be subject to review from time to time, resulting in a recommendation to:
   - leave as Released
   - update the current release with limited functionality (new issue)
   - update current release with patches (new version)
   - change the status to Withdrawn

2.5.2 Release numbering

1. A Release will be named (in the context of results/products produced by projects run by APM)
2. Software version control will manage all release numbering
3. A Release number consists of
   - release number [1-99]
   - issue number [0-99]
   - version letter [a-z]
   - status [alpha, beta, - , withdrawn]

2.5.3 Access to releases

1. A release is created in the software version control system
2. A release can be copied for the purpose of making tapes, disks, demonstrations etc.

2.5.4 Release version control

1. A release is created in the software version control system
2. Naming and numbering a release is decided on by the project manager of the project that creates the release
   - minor functional changes should result in an increase of the issue number by one
   - one or more patches applied to a release cause the version letter of the release to advance by one
3. A release is immutable until wthdrawn
4. A release which has status Withdrawn
   - should be archived to backing store
   - can be deleted after 2 years
5. The software version control system holds information on

— which modules make up this release
— what issue of each module applies
— what patches have been applied

3 Defect tracking

3.1 Definitions

3.1.1 Fault report

1. A fault report contains the following information:
   - number (allocated by APM)
   - name and version of product, release, module
   - time of occurance
   - description
   - frequency
   - context: hardware and software platform used:
   - Machine
   - Operating system name and version
   - Network type & software
   - Names and versions of other software directly supporting product (e.g. if the product is being used with a particular DBMS)
   - Names and versions of other software being run at the time
   - configuration
   - Installation options used
   - Configuration file contents
   - Relevant environment variables
   - compiler and linker info (version)

3.1.2 Bug report

1. A bug report contains the following information:
   - Name and version of faulty component
   - Name, organisation and contact information for developer
   - Estimated seriousness
   - Assigned priority, and id of budget holder.
   - Time form filled in.
   - Fault reports addressed by bug
   - Description of bug
   - List of affected source components (e.g. files, functions)
   - Estimate of effort required

— Description of fix

3.1.3 Change request

1. A change request contains the following information:
   — Name and version of faulty component
   — Name, organisation and contact information for developer
   — Assigned priority, and id of budget holder.
   — Time form filled in.
   — Fault reports addressed by change
   — Description of change
   — List of affected source components (e.g. files, functions)
   — Estimate of effort required

3.2 Procedures

3.2.1 Faults

1. Fault reports are received by APM and numbered upon receipt
2. Each fault report is archived and reviewed for action
3. The project manager decides whether a fault report is to be followed up
4. If so, the following actions are performed
   — Decide whether the behaviour described in the report is contrary to the specified behaviour of the system.
   — Attempt to reproduce the fault from the information described in the report.
   — Analyse the fault to find the bug in the software.
   — Search through outstanding fault reports to identify other manifestations of the same bug.
   — Examine what components and products are affected by the bug, estimate the seriousness of it and the resources required to fix it.
   — Fill in a bug report form. The bug is assigned an identifier which should be added to the fault reports which the bug relates to, and these fault reports should be marked and archived.

3.2.2 Bugs

1. Bug reports are reviewed periodically (triggered either by time or by number of bugs reports outstanding)
2. the project manager assigns priority to bug reports
3. the project manager assigns resources for bug fixes
4. the project manager seeks approval for changes to software (see change request)
5. For each bug the following actions are performed:
   — Notify owners of affected components.

- — Retrieve the necessary source code components from configuration control.
- — Modify the software and test it. Comments including the developer id and the reason for the change (change request id or bug id) should be included in the source. This is a sensible time for the change to be checked by another developer.
- — Measure and check any metrics required and perform regression testing.
- — Put the modified components into configuration control.
- — Update any obsolete documentation for the component, and components which depend upon it, to come into line with the changes made.
- — Notify the owners of all relevant components and products that the change has been made, so that they can make decisions about incorporating the modifications.

6. When the fix has been completed, it should be checked, signed off and archived.
7. Any software changes are to be made in a manner consistent with the procedures set out in the previous chapter.

3.2.3 Change requests

1. Change requests are requests for alterations to the specification of the system.
2. Change requests will be numbered for requirements traceability
3. Change request forms are reviewed periodically (triggered either by time or by number of Change request forms outstanding)
4. Changes to a release will result in a project to perform such changes

Note: This ought to be extended to cover new issues and versions too.

4 Tools

Although no decision can be made at this stage regarding a particular suitable tool set we feel it is meaningful to set expectations regarding the investment needed to equip the laboratory with up to date software tools.

4.1 Source code management systems

In a recent evaluation of 14 source code management systems[1], ClearCase from Atria came out as the favorite. It is supported on all our UNIX platforms. The evaluation did not mention PCs or other non-UNIX based operating systems). Claearcase licenses are $2500/seat and $800 maintenance per seat per annum.

Source code management systems also affect disk space requirements as would any quality system which guards versions of software or documents. Some systems are more hungry in this respect than others (TeamWare is more hungry than ClearCase for instance).

4.2 Defect tracking tools

The same recent evaluation selected DDTs 3.1.11 from Qualtrak at $700/seat as a suitable defect tracking tool.

We found a Web based project tracking tool at
<http://research.ivv.nasa.gov/projects/WISE/wise.html>
This would require an Oracle V6 database as back-end and has not yet been released.

4.3 Debugging and Testing

4.4 Memory management

We have Purify. This satisfies the requirement for memory leak detection etc. We would need to extend the license to more than the two seats we have now. It is not necessary to have a copy for each seat.

1. We refer to a confidential report which assessed 14 Source code management systems from as many companies in the context of a large scale development for a large consortium of clients.

4.5 Performance analysis

4.6 Development environment

References

[LINDEN 93]

van der Linden R. J., *An Overview of ANSA*; **AR.000.00**, APM Ltd., Cambridge U.K., May 1993.