SOFTWARE

QUALITY ASSURANCE

PLAN

LLNL YUCCA MOUNTAIN PROJECT

Revision 0
14 December 1989

LLNL YUCCA MOUNTAIN PROJECT

CONTROLLED
No. 044
THIS IS A
RED STAMP
LLNL
COPY

Software
Quality Assurance
Plan
14 December 1989

Author:	Nancyellen Cummins	14 December 1989
	Nancyellen Cummins	date
Approved:	Roger Aines	12/14/89
	Roger Aines Technical Area Leader	date
Approved:	David W. Short	12/14/89
	David Short Quality Assurance Manager	date
Approved:	Leslie J. Jardine	12/14/89
	Leslie Jardine YMPO-LLNL Technical Project Officer	date
Approved:		1/5/90
	YMPO Quality Assurance	date

117

Table Of Contents

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

APPENDIXES

FIGURE

TABLE

LIST OF EFFECTIVE PAGES

PAGE/SECTION NUMBER	CHANGE IN EFFECT	PAGE/SECTION NUMBER	CHANGE IN EFFECT
i through ix	Revision 0		
1.0	Revision 0		
2.0	Revision 0		
3.0	Revision 0		
4.0	Revision 0		
5.0	Revision 0		
6.0	Revision 0		
Appendix A	Revision 0		
Appendix B	Revision 0		
Appendix C	Revision 0		
Appendix D	Revision 0		
Appendix E	Revision 0		
Appendix F	Revision 0		
Appendix G	Revision 0		

LIST OF INCORPORATED CHANGE REQUESTS

CHANGE NUMBER	DATE	TITLE AND/OR BRIEF DESCRIPTION	SIGNATURE OF VALIDATING PERSON

1.0 PURPOSE

The purpose of this Software Quality Assurance (SQA) Plan is to establish the requirements for the planning, development, testing, tracking, documentation, and control of software used in support of the Lawrence Livermore National Laboratories (LLNL) Yucca Mountain Project (YMP) and to assign the responsibilities for implementing these requirements. The requirements are intended to ensure the control of software quality and to provide the Nuclear Regulatory Commission (NRC) staff with part of the basis on which they will evaluate the soundness of the physical and mathematical principles implemented within these software codes relative to their use to support license application to the same level of requirements as used to perform direct design analyses.

This plan uses guidance from NUREG-0856, and meets the requirements of LLNL QAPP, Section 3.0 and Appendix H. This plan is a revision controlled planning document. Changes to the SQAP or deviations therefrom will be submitted for Project Office approval.

2.0 APPLICABILITY

This plan describes the controls which address the requirements specified in QP 3.2, "Software Quality Assurance" and applies to software developed or used in support of analyses performed at Quality Assurance Level I and II. Although this Plan does not apply to analyses performed at QA Level III, it may be applied if desired.

For the purpose of this plan, software is categorized into three (3) types which are subdivided into two (2) groups. Table 6.1., "Software Categories", lists the types of software and the applicable sections of this plan. This plan states specific requirements for software in each of these types.

It may not be possible to meet the documentation requirements described in this plan for commercial software. Efforts must nevertheless be made to gather the information identified in this plan to aid in the use of the software and as proof of its credibility. The results of such efforts must be included within the required documentation.

Any modifications made to acquired, existing or commercial software that could affect the integrity of output must be planned, developed, documented, and tested under the same requirements as those for the in-house software in the same category. Configuration management requirements apply to acquired or commercial software.

The terms "software," "computer code," "code," "computer program," and "program" are used interchangeably throughout this plan. Databases are considered to be a type of software and are not discussed separately. Appendix A contains definitions of terms used throughout the plan.

3.0 RESPONSIBILITIES

This section outlines the responsibilities of the Technical Area Leader, Task Leader, Principal Investigator, Software Quality Manager, and Software Quality Technician. Upper levels of management recognize that SQA is a vital part of software development and that the individuals involved require their support to implement the actions of this Plan. This recognition by upper management must be translated into a commitment through policies that set software quality goals; establish SQA functions; and authorize the necessary resources to perform the tasks.

3.1 Technical Area Leader (TAL)

The TAL is responsible for the review and approval of the Individual Software Plan (ISP) [See Section 6.1.2 and Section 6.2.2].

3.2 Task Leader (TL)

The TL is the person responsible for actions pertaining to the development, correction, or change of a particular version of software. The TL understands the structure, control, and available options of the software (or will be able to find this information).

The TL will implement, where applicable, the software development life cycle. Most of the software documentation requirements will be written or created under the auspices of the TL, and the verification processes will generally be performed with his help. The TL will also decide when a software should receive a new version or release number.

3.3 Principal Investigator (PI)

The PI is responsible for a particular analysis or programming activity. The PI must ensure that all requirements for the software used in the analysis are met and that all documentation is in place for supporting analysis. It is the PIs responsibility to evaluate the impact of errors on previous calculations and determine the appropriate corrective action. It is the direct responsibility of the PI to oversee the software documentation required in Section 6.5 and to ensure that the certification process has been carried out to the extent appropriate to an analysis (see Section 6.6). The PI and the TL may be the same person.

3.4 Software Quality Manager (SQM) and Software Quality Technician (SQT)

The Software Quality Manager is assigned by the Deputy Project Leader and is responsible for overseeing implementation of this plan. The Software Quality Technician is assigned by the TAL responsible for the project and assists the Software Quality Manager. The duties that make up these tasks are described below:

3.4.1 The Software Quality Manager is familiar with this plan and will serve as consultant regarding its use when requested.

3.4.2 The Software Quality Manager is responsible for the collection or receipt of all documentation and written materials submitted to fulfill the requirements outlined in this plan.

3.4.3 The Software Quality Manager is familiar with the software development life cycle described in this plan and will serve as consultant regarding its use where required.

3.4.4 The Software Quality Manager is responsible for implementing, maintaining, and coordinating the configuration management system described in Section 4.0 of this plan.

3.4.5 The Software Quality Manager and the Software Quality Technician aid in preventing duplicate work from being performed by different PIs. For example, if more than one group is attempting verification of a particular model, they will make the groups aware of the other efforts.

3.4.6 The Software Quality Technician is responsible for working with the PI to determine whether the documentation collected for the software meets the requirements in this plan.

3.4.7 The Software Quality Technician is responsible for the storage of all documentation in accordance with the specification in Section 5.0, "Software Records Management System".

3.4.8 The Software Quality Technician ensures (with the records management personnel) that the storage of the formal released software product is maintained according to QP 17.0, "Records Management."

3.4.9 The Software Quality Technician ensures that documentation is completed in a timely manner and that records are complete.

3.4.10 The Software Quality Technician informs the TL when problems exist and maintains software records.

4.0 CONFIGURATION MANAGEMENT (CM) SYSTEM

As implemented at LLNL, software configuration management is the LLNL management process for: (1) assuring that the product is accurately described by the descriptive planning and product control documents; (2) assuring that changes are subject to a revision control process which provides comparable verifications and approvals as the original; and (3) providing status identification on the product, or on documentation traceable to the product.

The Configuration Management System described here is designed to assure positive identification of items (i.e. documents, programs, etc) or products and their baselines and for control of all products or item baseline changes. Configuration Management applies to Large and Medium SES and Calculational Non-SES. Configuration Management for categories of software which use notebooks is accomplished within the notebook (See Sections 6.1.1.A through 6.1.1.J).

A. Baselines

Configuration Management establishes two types of baselines. A baseline is a document or a set of documents containing a list of all configuration items or products formally designated and fixed at a specific time during a life cycle. A baseline, plus approved changes/updates specify the most current configuration identification for the item or product.

One type of baseline is the Configuration Item Baseline and the other is the Software Product Release Baseline. The former is established when the item is initially submitted to Configuration Management, while the latter is established when the software product is (formally) released. The Software Quality Manager receives documentation and software from the TL during the software life cycle. The Software Quality Manager records receipt of the original software or documentation in the corresponding baseline document; all subsequent versions or revisions to the originals constitutes updates to the original baseline. The software Product Release Baseline is the sum of all the Configuration Item Baselines that constitute the Software Product at release. A Master Log is kept of both baselines.

B. Notebook

The Notebook will provide a path so that the reviewer can follow the work that is being done or was done. A Configuration Item Identifier (CII) will be assigned each item, as described in section 4.1.1, and be recorded along with version numbers and program names in the Notebook. Error Resolution and change requests will also be documented within the Notebook.

The Software Quality Manager is responsible for the initiation, implementation, and maintenance of the Configuration Management procedure(s) and the Configuration Management system. Configuration Management is a system of controls, verifications, approvals, and authorizations which provide change control and traceability for software and documentation. Configuration Management also provides a means for error reporting, resolution, and tracking.

Configuration Management documentation when completed is submitted to the Software Records Management System for archiving.

4.1 Configuration Identification

A configuration item baseline is established at the completion of each phase of the software development cycle. Elements of the software product placed under Configuration Management are called configuration items, and each is assigned a unique identifier. Items so identified, include software and documentation, are listed in the Configuration Item Baseline document, and are not changed without authorization. Approved changes to a baseline are added periodically to the baseline as updates. Updates are incorporated into subsequent baselines. Both baselines and updates are defined by their composition of software configuration items.

A labeling system for configuration items will:

* Uniquely identify each configuration item.
* Identify changes to configuration items by revision, version, change, or release number.
* Placed the configuration item in a relationship with other configuration items.
* Provides the ability to reconstruct the configuration of the software or documents from the requirements document to the present time.

A software product ready for release is assigned a unique identifier and is placed on the Software Product Release Baseline, and is not changed without proper authorization.

Each Configuration Item is assigned an identifier. That item is baselined. The Item Baseline consists of the original item and its update(s) (e.g. version number for software, revision number for documentation).

Each product has a release identifier. When a product is completed, that product is the sum of all the configuration item identifiers (because it is the sum of all the configuration item baselines for that product). The baselines can be cross-referenced from item to product and vise-versa.

Section 4.1.3, "Identifier Examples", shows one method for tracking changes to an original item or product. The implementation details will be in a TIP.

4.1.1 Configuration Item Identifier (CII)

The Software Quality Manager receives from the TL the review reports, documentation, and/or software at the completions of each phase. Each document or software received is assigned a unique Configuration Item Identifier and is placed on the Configuration Item Baseline.

4.1.2 Release Identifier

The Software Quality Manager assigns release identifiers and monitors all released software products. The initial software product release is identified by a unique release identifier. This release identifier appears on all items of the software product (i.e. software and documentation) and is updated at each subsequent release. The Software Product Release Baseline is also updated to reflect the change.

4.1.3 Identifier Examples

This section describes a numbering system for identifiers, both Item and Release, and their changes. Each document or program is given a unique alpha-numeric item identifier by the Software Quality Manager upon entering configuration management. Also a unique alpha-numeric release identifier is assigned at the time of product release. The item identifier is composed of four (4) sections: Type, Descriptor, Numeric Sequence, and Status.

Where: *TYPE* is either "D" for document or "P" for Program.
DESCRIPTOR is an abbreviation of the item (i.e. "TM" for the Theoretical Manual or "LP" for Library Program).
NUMERIC SEQUENCE is a number that designates the order received.
STATUS shows the state of the item (i.e. "blank" for Original; "V" for version; "R" for revision; and "C" for change).

Using the above description the following identifier would be assigned to a Theoretical Manual which was the fifteenth item received: *D-TM-015* (no status designation used as this is an original). The following identifier would be assigned to an original program which was the sixteenth item received: *P-LP-016* (no status designation used as this is an original).

Status indicators establish the state of the item. The following are examples of a numbering system:

Item Identifiers:

Status indicator for Documents:

D-TM-015.C001 (First change to Theoretical Manual item number 15)
D-TM-015.C005 (Fifth change to Theoretical Manual item number 15)
D-TM-015.C0010 (Tenth change to Theoretical Manual item number 15)

D-TM-015.R001 (First revision to the Theoretical Manual item number 15)

D-TM-015.R001.C001 (First change to First revision to the Theoretical Manual item number 15)
D-TM-015.R001.C005 (Fifth change to First revision to the Theoretical Manual item number 15)
D-TM-015.R001.C0010 (Tenth change to First revision to the Theoretical Manual item number 15)

D-TM-015.R002 (Second revision to the Theoretical Manual item number 15)

Status indicator for Programs:

P-LP-016.C001 (First change to Library Program item number 16)
P-LP-016.C005 (Fifth change to Library Program item number 16)
P-LP-016.C0010 (Tenth change to Library Program item number 16)

P-LP-016.V001 (First version to Library Program item number 16)

P-LP-016.V001.C001 (First change to first version Library Program item number 16)
P-LP-016.V001.C005 (Fifth change to first version Library Program item number 16)
P-LP-016.V001.C0010 (Tenth change to first version Library Program item number 16)

P-LP-016.V002 (Second version Library Program item number 16)

The Release Identifier is composed of four (4) sections: Type, Descriptor, Numeric Sequence, and Status.

Where: *TYPE* is "PR" for "Product Release".
DESCRIPTOR is an abbreviation of the product name (i.e. the activity name).
NUMERIC SEQUENCE is a number that designates a numeric value established by the activity.
STATUS shows the state of the item (i.e. "blank" for Original; "R" for revision; and "C" for change).

Using the above description the following identifier would be assigned to a Product Release which has some "name" and "value": *PR-name-999* (no status designation used as this is an original).

Status indicators establish the state of the product release. The following are examples of a numbering system:

Release Identifiers:

Status indicator for Releases:

PR-name-999.C001	(First change to release identifier)
PR-name-999.C005	(Fifth change to release identifier)
PR-name-999.C0010	(Tenth change to release identifier)

PR-name-999.R001	**(First revision to the release identifier)**
PR-name-999.R001.C001	(First change to First revision to the release identifier)
PR-name-999.R001.C005	(Fifth change to First revision to the release identifier)
PR-name-999.R001.C0010	(Tenth change to First revision to the release identifier)
PR-name-999.R002	**(Second revision to the release identifier)**

4.2 Configuration Change Control

Configuration Change Control provides the controls necessary to manage and control the change process. Changes to the software product or configuration item are accomplished through implementation of an Error Reporting and Change Request Tracking (ER/CRT) System. Changes to software or documentation are subject to the same level of approval, review, and verification as the original software or documentation. The ER/CRT system allows personnel working with the software or documentation to suggest a change to the software's capabilities or to submit a request for error correction. Normally changes are requested to accomplish at least one of the following:

* Correct errors.

* Implement new or revised requirements.

* Change testing or design input parameters.

* Add data, results, or functions.

* Remove out-dated information.

The Software Quality Manager in accordance with the Configuration Change Control process assures that approved changes are documented and incorporated; assures that changes are assessed for their validity; assures that all changes are approved; monitors the change status; and informs the TAL, TLs, PIs, developers, and Registered Users of the error correction or change status.

The Software Quality Technician will maintain a list of all open (unresolved) Error Reports (ERs) and distribute a list of these along with closed (resolved) ERs on a quarterly basis. Also the Software Quality Technician maintains a list of all open and closed Change Requests (CRs) and distributes the list on an annual basis. The distribution of this list will include the TLs, TALs, QA manager, Project leader, ER or CR originators.

4.2.1 Registered User Distribution Log

LLNL-YMP will establish a Registered User Log for all Quality Level I and II software user. The Log will contain the institution/organization name, address, telephone number(s), and the primary contact of the user. Registered Users will receive all error/change resolution documents.

4.2.2 Error Report/Change Request Tracking (ER/CRT) System Procedure(s)

ER/CRT system procedure(s) address the process of correcting errors or making changes, including additions of code in accordance with Section 6.0, to the configuration item or software product.

ER/CRT System Procedure(s) include steps for identifying discrepancies in writing, documenting proposed changes, identifying the originating organization, providing the rationale for the change, reviewing proposed changes for adequacy, reverifying the affected software, identifying affected baselines and software configuration items, obtaining approval and authorization for correction/change, and verifying that the correction/change has been incorporated. Procedures also identify an accepted time frame for the change to take place. The correction/change is formally evaluated by a qualified individual or organization with the ability to approve or disapprove the proposed correction/change. Assurance is provided that only authorized corrections/changes are made to the configuration item or software product baselines.

ER/CRT system procedure(s) establish a mechanism for feedback to the users for information about specific problems and recurrent types of problems. In addition, the procedure requires users (in-house) to inform the TL responsible for the product or configuration item when errors are discovered. The TL can examines and assesses the overall effect of the errors on the code. Users are provided with sufficient information to determine what effect the errors or changes have had on previous calculations, applications, or decisions.

The TL is ultimately responsible for the resolution of errors discovered during development or use and for making recommendations for changes. The TL assesses the impact of noted errors or changes requested and decides if the impact of the resultant change will be minor or significant. A significant change requires reverification. After the significance of the error or change is assessed, the TL informs all Registered Users of the action planned and the effect of the correction/change on the results already obtained with the program. Notification of users will be accomplished in accordance with QP 3.5, "Control of Internal Technical Interfaces". The assessment of the defects and impact on previous applications is the responsibility of the Registered User. The assessment of the defects and impact on previous applications is the responsibility of the Registered User.

4.3 Configuration Status Accounting and Reporting

The purpose of Configuration Status Accounting is to develop and maintain records of the status of the software product or configuration item as it moves through the software life cycle and to track software applications used in support of level I and II analyses. The status of the current baselines including pending, approved, and implemented changes are recorded. A detailed record of the software product configuration is maintained and made available to assure that the overall configurations are effectively managed. The Master Log is the means by which this accounting and reporting is accomplished.

Configuration Management monitors content, format, author, code developer, distribution, and timing of reports. The following information may be obtained from the configuration records:

* Current baseline list of configuration items.
* Latest revision and change information for baselined documents, including pending or approved changes.
* Latest release, version, and change information for baseline software, including pending or approved changes.
* Audit trail for each configuration item detailing changes of each baseline during the software product's life cycle.
* Hierarchy of documents.
* Distribution, documentation, and change control information.
* Report on completion of test suite by latest version, and all previous versions, of the software.
* Latest version of and change information on test suite for a software product.
* Test suite and verification libraries used for testing.
* Applications and their status.

4.3.1 Master Log

The master log will establish traceable records regarding new versions, new releases, and Level I and II applications. The Log will contain the following:

a. Each unique version and release will be identified on its own page in the log. This page will identify the name of the software, the version and release number, the date of entry into the Configuration Management system, Configuration Item Identifier number, the name of the person releasing the item, and the name of the TL. Each use of the software for a QA Level I or II analysis, or each transmittal of the software to any group intending QA Level I or II analysis, will be identified on this page by a unique and traceable designator such as product number, change number, activity number, or WBS number.

b. Each application will be identified on its own page in the log. This page will identify the application/analysis to be performed, name of the software to be used, the software version and release number, and verification status.

c. Within the log, the most current version of the software will be identified so that the Software Quality Manager can direct new users of the software to the appropriate TL.

d. The information maintained in the log will be provided on the Software Summary (see Appendix E), and by material submitted in support of verification and analysis efforts.

e. When not in use by the Software Quality Manager, the master log will be stored in the Software Records Management System in a location setup by the Software Quality Technician.

5.0 SOFTWARE RECORDS MANAGEMENT SYSTEM (SRMS)

The Software Records Management System is a system for controlling, storing, and providing access and security for documents and software. The following subsections provide a general description of what is required for the Software Records Management System. Detailed description and implementation are provided by TIPs.

5.1 Documentation Storage

The Software Records Management System will be used to store all software product material, making it readily available and easy to locate. The Software Quality Manager is responsible for carrying out the requirements of Software Records Management System, providing material to the Local Records Center (LRC) at required times, for making modifications to procedures as needed and providing modified procedures to the central system.

5.1.1 All documentation required by this plan will be stored within the Software Records Management System by the Software Quality Technician until transmitted to the Local Records Center.

5.1.2 Software and documentation will be assigned a unique identifier for storage and tracking purposes.

5.1.3 Each document or software that is tracked in the Configuration Management system (see Section 4.0) will have a file for each unique version. This file will contain all the applicable documentation defined in 6.1 through 6.5, or references to it. Different releases of the same software version or different revision of a document will also be stored in that same file but with a clearly marked separator.

5.1.4 Each analysis effort will have its own unique file location, cross-referenced to the file containing the software and the documentation for the version of software it used. This file will contain the documentation defined in 6.6 through 6.9, or references to its location.

5.1.5 Documents may be referenced to in more than one location but need not be duplicated between the different files. The Software Quality Technician will store one copy of the document and make certain that all other references to it are easily traceable to its location.

5.2 Media Control and Storage

The control of physical media, access authorization, and security are the functions that assure that the stored software is physically retrievable and cannot be lost or compromised by day-to-day operations or catastrophic events. Typical storage media includes magnetic disks and diskettes, magnetic tapes, and paper hard copy listings.

5.2.1 Physical Media

The physical media upon which the software is stored is controlled so that the software is not damaged, altered, or degraded. This is accomplished by providing adequate safe storage techniques and by software and data backups.

5.2.1.1 Backups

At least two (2) backup copies are produced for data and software. The number and frequency of backups are to be established. The frequency of the backups should be set to try to minimize the amount of reworking required by the developers. Backups are stored in a suitable environmentally controlled and secure location. Two copies of each backup are made, one is stored locally and one is stored at a remote location. The Software Quality Technician is responsible for instituting a Software Backup Log, Schedule, and for initiating the backups. The Backup Log contains the following information: date; storage media; medium number; file identification; and the physical location of the copies. The Backup Log becomes part of the software product.

5.2.2 Software Librarian

The Software Quality Technician is the software librarian and is responsible for assuring that only the approved versions are distributed and used for analysis, and that any software changes are made in accordance with established procedures.

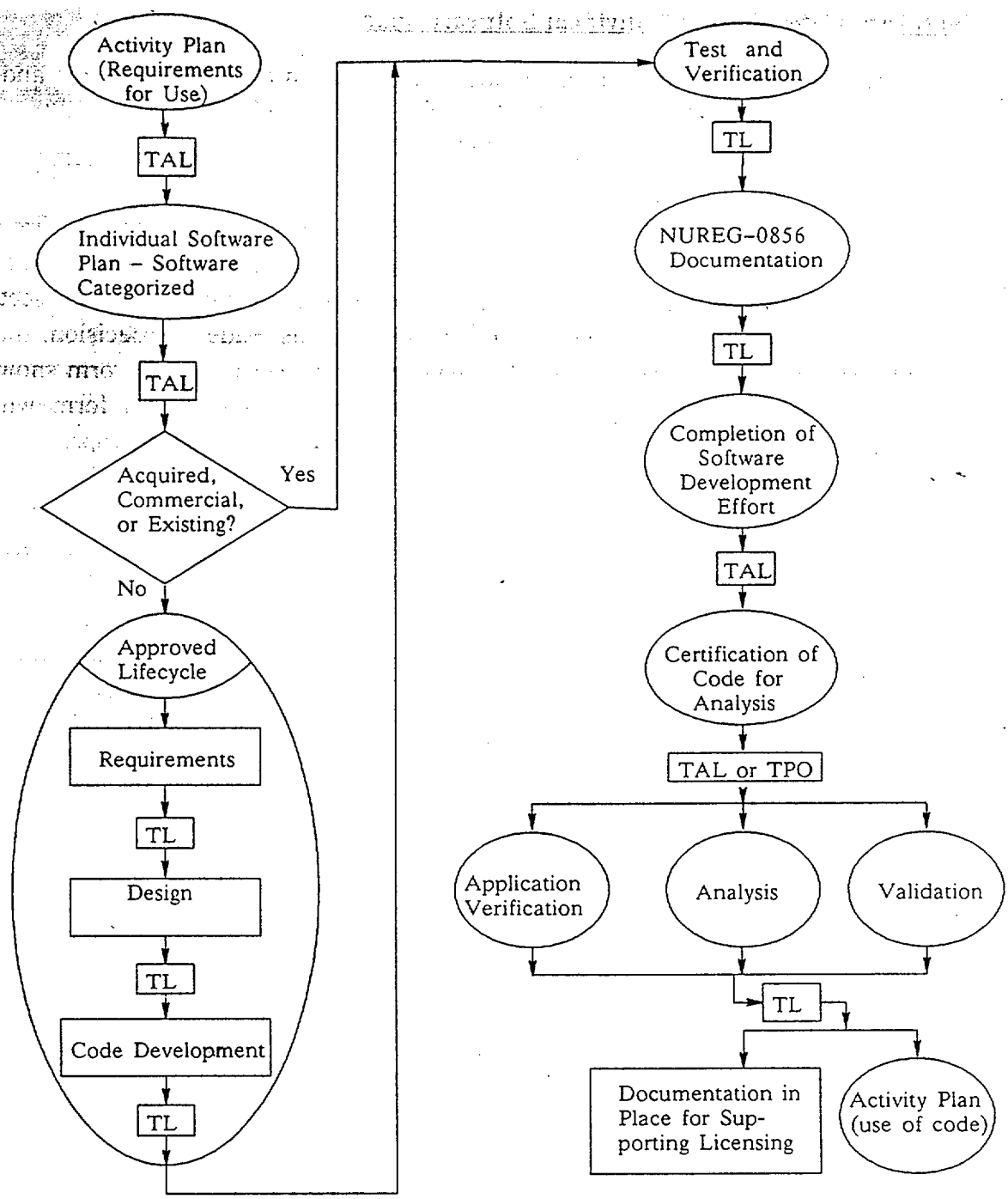
6.0 REQUIREMENTS

This section specifies the requirements for software planning, development, and use. A flowchart of the software process is shown in Figure 6.1, "Software Process Flowchart". Sections 6.1 through 6.8 describe all the steps in planning, development, testing, and use. Section 6.9 describes software product completion and post-release maintenance. Table 6.1, "Software Categories" describes, in brief, the software categories covered by the plan and sections 6.1.1.A through 6.1.1.J specifies the detailed requirements that each category of software will meet.

This plan is provided as a basis for the content of all required documentation. Detailed documentation requirements are described in Appendix B. Format, arrangement, and style are left to the discretion of the writer as long as all necessary topics are addressed. If a portion or all of the information called for is already written and located in some other document, that document may be submitted with the necessary topics identified and need not be rewritten. Comment lines within a source code listing shall be added to enhance appropriate portions of documentation called for in the plan.

A checklist showing the detailed requirements for Scientific and Engineering Software (SES) documentation is included in Appendix D. A copy of the checklist should accompany each document or review submitted to the Software Quality Manager for permanent storage.

Depending on the software category, documentation described here must be subjected to a technical review before it becomes part of a software product's permanent file. These reviews must be in accordance with the requirements set forth in QP 2.4, "Technical Review". A copy of each technical review report will be stored by the Software Quality Manager in the Software Records Management System.



TAL = Technical Area Leader Review/Approval Required
TL = Task Leader Review/Approval Required
TPO = Technical Project Officer Review/Approval Required

Figure 6.1. Software Process Flowchart.
(Reference Life Cycle and Development Process)

6.1 Software Category and Individual Software Plan

The software category selection and individual software plan are required items and are completed prior to the start of any work.

6.1.1 Software Category

The first step of the Software process is to assess what category the software falls under. Software is divided into three (3) types and each type is subdivided into two (2) groups. Table 6.1., "Software Categories," lists all the applicable sections that each type must complete. After the TL or PI has made this decision, the TL or PI will submit a memo to the TAL regarding this decision. The form shown in Appendix C. "Software Category Selection" will be used. This form will be approved by the TAL under whose WBS this work is funded. This approved form is a precursor to the Individual Software Plan (see section 6.2.2). The approved form will be sent to the Software Quality Manager and stored in a file created for this software within the Software Records Management System. The Software Quality Manager or Software Quality Technician will send a copy of the approved form to the responsible TL or PI after the form has been assigned a Configuration Item Identifier and logged. The software type selection form is required for large and medium SES, large Calculational Non-SES created in-house, acquired, existing, or commercial Calculational Non-SES which will be changed in-house, and Non-Calculational software created in-house. It is the TL's responsibility to ensure that the Software Category Selection form is filed, with the Software Quality Manager, for the required software as defined here. Listed in sections 6.1.1.A through 6.1.1.J are the detailed requirements for each software category.

	In-house Software (sponsored by YMP)	Acquired, Existing, and Commercial Software
SES	Large SES (e.g. EQ6, Pandora, Tough). * Multiple person efforts greater than 5000 lines or 6 months. - ISP is required - Full life cycle required. - Life cycle documents stand alone and are under configuration management. Medium SES * Efforts of less than 5000 lines or 6 man months. - All one-person efforts greater than 500 lines and less than 5000 lines long are Medium SES. - Life cycle required (notebook may be used). - ISP required. - NUREG-0856 documentation required. - Configuration Management required. Small SES * Less than 500 lines - NUREG-0856 and Life cycle documentation required (Notebook may be used). - Controlled under activity plan by TL as appropriate for activity.	Large SES. * Includes all codes larger than 500 lines. - Handled same as in-house SES following acquisition (e.g. configuration management). - ISP and Acquisition plan required. - Verification required. - Available documentation of development obtained. - NUREG-0856 documentation required. [Medium SES] - *No medium SES category is defined for acquired, existing, and commercial software.* Small SES * Same definition as in- house small SES. - Handled same as in- house once acquired. - Acquisition controlled under normal purchasing controls. - NUREG-0856 documentation required (Notebook may be used).
Calc. Non SES	Large Calculational Non-SES * Large efforts greater than 500 lines - ISP required - Life cycle required (notebook may be used) - NUREG-0856 Documentation required. - Configuration management required. Small Calculational Non-SES * Less than 500 lines. - Controls as specified in activity plan when critical to QA I or II work.	All Calculational Non-SES - Configuration management if changes are to be made. - Available documentation must be sufficient to determine suitability for use. - Acquisition controlled under normal purchasing controls.
Other	Non Calculational Software - Controls specified in activity plan. - ISP required when identified in activity plan.	Non Calculational Software - Controls specified in activity plan. - ISP required when identified in activity plan. - Acquisition controlled under normal purchasing controls.

Table 6.1 Software Categories

Note:* "Lines" are to be considered single instructions, exclusive of comments or other descriptive material.

6.1.1.A Large In-House SES (or Large SES sponsored by LLNL-YMP).

Scientific and Engineering Software is software that specifies operations according to a physical or mathematical model or that uses a numerical method and supplies primary data or analysis used in support of Level I and II activities. This category includes scientific, engineering, and mathematical modeling software that use a numerical method. Typical functions for this software include geochemical models, repository or waste package performance assessment, safety and reliability studies, engineering design, etc. Commercial software packages applied to these types of activities are included. This category includes mathematical subroutine libraries used in SES.

Large SES is long and written by multiple authors. This necessitates detailed planning and controls on implementation. Large SES is: any SES software written by more than one person which is more than 5000 lines (instructions) long or which requires more than six (6) man-months to code.

1. Category/ISP - Required.

2. Life Cycle - Full life cycle implementation, with separate documents for requirements, design, verification, and validation summary. Individual modules may have their own life cycle components. Rapid prototyping may be part of development cycle.

3. Configuration Management - Full management of coding and life cycle documents.

4. Documentation - Full life cycle documents, preferably as UCID's where feasible. Full NUREG-0856 documentation prior to software release, with exception of validation summary (provided later).

5. Certification - May be certified for certain uses after the Test, Debug, and Verification Phase.

6. Verification - Plans for verification to be included, or referenced for future development, in ISP. Verification fully documented and results reviewed under document review rules. Planning for verification must be documented with results to ensure adequate review.

7. Validation – Validation efforts will be documented in published reports, reviewed by document review methods. Validation plans for work conducted by LLNL specifically for validation typically will be in a SIP and Activity Plan, but may be addressed in the Individual Software Plan. Other work may also be included in validation reports. Summary or compilation of all validation efforts will be provided at licensing time as part of "code assessment and support" clause of NUREG-0856.

8. Completion – Completion Memo is required (progress will be logged through the use of the master log; when all requirements have been met, the completion memo will be filed).

6.1.1.B Medium In-House SES

Medium SES is shorter than Large SES, or is written by a single individual. This permits scientific notebooks to be used in the documentation of the coding activities.

Medium SES is written by only a single individual (and is greater than 500 lines and is less than 5000 lines); or when written by more than one person is less than 5000 lines long or requires less than six (6) man-month to code.

1. Category/ISP - Required: the TAL must sign to ensure that effort is actually in this category. Should the effort grow into large category, all requirements for Large SES must be met. This category is intended to provide reasonable control of development, and adequate documentation that if the effort should grow, the large SES requirements could be met by preparing the documents using the notebook records.

2. Life Cycle - Life cycle must be met, with the notebook containing the information sufficient for review of the product. Notebooks are subject to the normal notebook review and verification procedures.

3. Configuration Management - Required (notebook records may be used).

4. Documentation - Life cycle documentation is contained in the notebooks (may be in a number of locations in the notebook but must be complete). Full NUREG-0856 documentation required, with Individual Software Plan defining what manuals will be written (in most cases software of this size will only have one document describing them).

5. Certification - May be certified after the Test, Debug, and Verification phase.

6. Verification -Required (Notebooks may be used).

7. Validation - Recorded in notebook or as document (preferred), validation planning must be approved as for large SES, but may be carried out and recorded in a notebook by TL approval.

8. Completion - Same as large SES.

6.1.1.C Small In-House SES

Small SES is less than 500 lines long and is verifiable by direct inspection. It may readily be documented by the use of comments written within the software. Additional controls may be required in the activity plan.

1. Category/ISP – This software must be recorded and controlled as specified in the activity plan. The TAL must approve the classification to ensure that the effort is actually in this category. Should the effort grow into a medium or large category, all requirements for Medium or Large SES must be met. This category is intended to provide reasonable control of development, and adequate documentation that if the effort should grow, the medium or large SES requirements could be met by preparing the documents using the notebook records.

2. Life Cycle – Life cycle must be met, as specified in the ISP, with the notebook containing the information sufficient for review of the product. Notebooks are subject to the normal notebook review and verification procedures .

3. Configuration Management – Must have unique identifier attached.

4. Documentation – Life cycle documentation is contained in the notebooks (may be in a number of locations in the notebook but must be complete). Full NUREG-0856 documentation is required, with the activity plan defining what manuals will be written (in most cases software of this size will only have one descriptive document).

5. Certification – Must be certified for use in Level I or II analysis by review for suitability if activity plan identifies this software as critical to the Level I or II analysis (if it is the principal software used in the analysis, for instance). This type of software should be primarily controlled under the activity plan.

6. Verification – Direct inspection (Software which is not verifiable by inspection should be handled as Medium SES).

7. Validation – Only required as per an activity plan.

8. Complete – Does not apply, unless specified in the an activity plan.

6.1.1.D Acquired, Existing, or Commercial Large SES

Acquired, Existing, or Commercial Large SES are equivalent to the large and medium in-house SES categories. (Notebook control is not used for acquired, existing, or commercial software).

1. Category/ISP - Required, submitted with acquisition plan (which serves as ISP for acquired software).

2. Test, Debug, and Verification - As specified in the ISP.

3. Configuration Management - Must have unique identifiers. If the software is to be modified or used at LLNL, it must be placed under configuration management. In general the version number on a commercial product will be sufficient.

4. Documentation - Available documentation regarding development is to be obtained. Acquired software must meet NUREG-0856 documentation requirements either from vendor documents or LLNL supplements. If the software as acquired does not meet the NUREG-0856 documentation requirements, the acquisition plan or an update thereto will describe the plan to meet the requirements.

5. Certification - Same as in-house SES.

6. Verification - Acquisition plan describes the verification activities to be performed by the vendor as part of the purchase agreement and any verification activities to be conducted by LLNL. Results of the verification activities are to be substantially the same as for in-house SES of the same size.

7. Validation - Same as in-house SES

8. Complete - Same as in-house SES

6.1.1.E Acquired, Existing, or Commercial Small SES

Small Acquired, Existing, or Commercial SES is less than 500 lines long and is verifiable by direct inspection. It may readily be documented by the use of comments written within the software. Additional controls may be required in the activity plan.

1. Category/ISP – This software must be recorded and controlled as specified in the activity plan. An Acquisition Plan will be submitted (which serves as an ISP for acquired software).

2. Test, Debug, and Verification – May be recorded as part of activity records as appropriate.

3. Configuration Management – Must have unique identifier.

4. Documentation – Available documentation regarding development is to be obtained. Acquired software must meet NUREG-0856 documentation requirements either from the vendor documentation or LLNL supplements. If the software as acquired does not meet the NUREG-0856 documentation requirements, the acquisition plan or an update thereto will describe the plan to meet the requirements.

5. Certification – Same as in-house SES.

6. Verification – Required as defined in an activity plan. Software that is not verifiable by inspection should be treated as large acquired SES.

7. Validation – Required as defined in an activity plan when critical to the analysis.

8. Complete – Does not apply, unless specified in an activity plan.

6.1.1.F Large In-House Calculational Non-SES

Large In-House Calculational Non-Scientific and Engineering Software (SES) is greater than 500 lines. Calculational Non-SES is software that can contain complex mathematics, but does not involve modeling of scientific or engineering investigation or design applications. Some examples include statistical packages, graphics packages, mesh generators, spread-sheet programs, software operating analytical instruments, embedded codes, etc.

All efforts of more that 500 lines are in this category.

1. Category/ISP - Required.
2. Life Cycle - Life cycle must be met, with notebook containing the information sufficient for review of the product. Notebooks are subject to the normal notebook review and verification procedures.
3. Configuration Management - Required (notebook records may be used).
4. Documentation - Life cycle documentation is contained in notebooks (may be in a number of locations in the notebook but must be complete). Full NUREG-0856 documentation required, with Individual Software Plan defining what manuals will be written.
5. Certification - May be certified after Test, Debug, and Verification phase.
6. Verification - Required (Notebooks may be used).
7. Validation - Does not apply, unless specified in the ISP.
8. Complete - Completion memo is filed at end of effort.

6.1.1.G Small In-House Calculational Non-SES

Small In-House Calculational Non-SES is less than 500 lines. Calculational Non-SES is software that can contain complex mathematics, but does not involve modeling of scientific or engineering investigation or design applications.

1. Category/ISP - Does not apply. This software does not have to be recorded or controlled unless specified in the activity plan as critical to a level I or II analysis (then it must be certified for use).
2. Life Cycle - Does not apply, unless specified in an activity plan.
3. Configuration Management - Must have a unique identifier attached.
4. Documentation - Documentation sufficient for review for use should be included in the source code. No other documentation required.
5. Certification - Must be certified for use in Level I or II analysis by review for suitability, if activity plan identifies this software as critical to the Level I or II analysis. This type of software should be primarily controlled in the activity plan.
6. Verification - Software which is not verifiable by inspection should be handled as large Calculational Non-SES.
7. Validation - Only required as per an activity plan.
8. Complete - Does not apply, unless specified in an activity plan.

6.1.1.H Acquired, Existing, or Commercial Calculational Non-SES

Acquired, Existing, or Commercial Calculational Non-SES is software that can contain complex mathematics, but generally does not involve modeling of scientific or engineering investigation or design applications.

All Acquired, Existing, or Commercial Calculational Non-SES is handled similarly.

1. Category/ISP - Does not apply. This software does not have to be recorded or controlled unless specified in the activity plan as critical to a level I or II analysis (then it must be certified for use).

2. Test, Debug, and Verification - May be recorded as part of activity records as appropriate.

3. Configuration Management - Must have unique identifiers. If the software is to be modified or used at LLNL, it must be placed under configuration management. In general the version number on a commercial product will be sufficient.

4. Documentation - Available documentation regarding development is to be obtained. Available documentation must be sufficient to determine suitability for use.

5. Certification - As specified when identified in an activity plan.

6. Verification - Activity plans or purchasing documents describe verification activities to be performed by the vendor as part of purchase agreement and any verification activities to be conducted by LLNL.

7. Validation - As specified when identified in an activity plan.

8. Complete - As specified when identified in an activity plan.

6.1.1.I In-House Non-Calculational Software

Non-Calculational software is software that performs data or symbol manipulation but not mathematical calculations. Examples include compilers, word processors, operating systems, etc.

All Non-calculational software is handled similarly.

1. Category/ISP - Required when specified in the activity plan.
2. Life Cycle - Does not apply, unless specified in an activity plan.
3. Configuration Management - Must have a unique identifier attached.
4. Documentation - Documentation sufficient for review for use should be included in the source code. No other documentation required..
5. Certification - Must be certified for use in Level I or II analysis by review for suitability, if activity plan identifies this software as critical to the Level I or II analysis. This type of software should be primarily controlled under the activity plan.
6. Verification - When identified in an activity plan.
7. Validation - Only required as per an activity plan.
8. Complete - Does not apply, unless specified in an activity plan.

6.1.1.J Acquired, Existing, or Commercial Non-Calculational Software

Non-Calculational software is software that performs data or symbol manipulation but not mathematical calculations.

All Non-calculational software is handled similarly.

1. Category/ISP - Does not apply. This software does not have to be recorded or controlled unless specified in the activity plan as critical to a level I or II analysis (then it must be certified for use).
2. Test, Debug, and Verification - Does not apply, unless specified in an activity plan.
3. Configuration Management - Must have a unique identifier attached.
4. Documentation - Available documentation should be acquired, but need not be supplemented unless inadequate to review for suitability for use.
5. Certification - If critical, handle as In-House Non-Calculational.
6. Verification - Required, when defined in purchasing documents.
7. Validation - Does not apply, unless specified in an activity plan.
8. Complete - Does not apply, unless specified in an activity plan.

6.1.2 Individual Software Plan (ISP)

Individual Software Plans are required before beginning software related work. Individual Software Plans describe the methods to be used in complying with the SQA plan, and to conform with the requirements of activity plans (See QP 3.0, Appendix A - "Suggested Content and Format for Activity Plans") for software development activities. Individual Software Plan's are required for Large SES (in-house and acquired), Medium SES (in-house), Large Calculational Non-SES (in-house), and Calculational Non-SES (acquired) that will be changed at LLNL. For acquired Large SES, the Individual Software Plan contains the plan for acquisition of the software. For existing software, the Individual Software Plan describes how the requirements of this plan will be met. Individual Software Plans are approved at LLNL by the TAL with the concurrence of the Software Quality Manager using the same mechanisms as activity plans (See QP 3.0.8, "Review and Approval").

Individual Software Plans describe software controls which arise from requirements of the SQA plan and any additional requirements that arise from the TALs review of the activity plan. The TAL will specify additional controls for critical software. These additional controls will arise particularly for small categories of software which are critical to a specific scientific investigation. The uses of software of any category are subject to verification in the course of technical review of the scientific activity.

When software development is a large activity, the Individual Software Plan may contain the information required for the activity plan as well as software information. The Individual Software Plan may also only address software development in conjunction with an existing activity plan.

The specific material regarding software required in the Individual Software Plan is:

a. Software Category. Sufficient information should be included that the TAL may evaluate the final use and condition (i.e. size) of software. Module additions to existing codes retain the category of the original software even if the module is smaller in size.

b. Life Cycle. Describe how the reference life cycle will be implemented. (See Section 6.2). Describe hold points and products which require approval of TL. The ISP addresses implementation of the SQAP requirements.

c. Test, Debug, and Verification. Describe for acquired, existing, and commercial software.

d. Configuration Management. Describe the configuration management system to be used (e.g. Configuration Management will be accomplished in accordance with Section 4.0 of the SQAP.).

e. Documentation. Describe the documentation to be provided at the completion of each phase and when work is finished.

f. Certification. If certification for use is planned prior to completion of the development, describe the anticipated use and project condition at that point.

g. Verification. Describe plans for verification in general terms.

h. Validation. Describe plans for validation in general terms or provide a reference to the documents (e. g., SIP, Activity Plan) describing these plans.

Individual Software Plan's are to be applied to software in such a way as to provide easily reviewed packages of software. This may be individual software, packages of software, or codes and database. The TAL determines appropriate Individual Software Plan packaging.

6.2 Software Development (Reference) Life Cycle

Software development (reference) life cycle is a process by which software is planned and developed in phases, with a specific product or output, associated documentation and review at each phase. The goals of the life cycle process are: 1) to increase quality of the product and productivity of the personnel; 2) establish traceability throughout the process; and 3) establish a good documentation base for critical software. The life cycle process provides continuity for programming efforts so extensive that the same personnel may not be working on the effort throughout its life.

6.2.1 When to Use Life Cycle

Software development is under the life cycle process when the programming effort has not begun or when acquired, existing, or commercial software needs modification or error correction. Life cycle is required as specified in Table 6.1, "Software Categories". Software which was originally categorized as not requiring life cycle will later require life cycle if during development the software changes classification or size and the new classification or size requires life cycle. The TL or PI is required to meet all life cycle requirements as specified by the new category. The criteria for deciding the level of detail to be used during the life cycle process are based on the size and criticality of the programming effort and are specified in this plan.

If the software does not meet the criteria established in Table 6.1 the TL or PI must write a memo that documents this decision. This memo must be approved, prior to the beginning of coding, by the responsible TL or TAL and the QA manager. The approved memo will be sent to the Software Quality Manager and stored in the file created for the software within the Software Records Management System. The Software Quality Manager or Software Quality Technician will send a copy of the approved memo to the responsible TAL, TL and PI.

6.2.2 Individual Software Plan (ISP)

The reference life cycle process is shown in Section 6.2.3, "Life Cycle Phases". If the software effort requires the use of life cycle those portions of Section 6.2.3 that are applicable to the effort must be performed. The life cycle or portion of the life cycle to be used must be planned out in advance by the PI or TL and specified in the Individual Software Plan. The Individual Software Plan is a precursor to starting any work (i.e. software requirements specification, other documentation or coding). The approved plan will be sent to the Software Quality Manager who will enter the plan in the Configuration Management system and store it in a file created for the software within the Software Records Management System. The Software Quality Manager or Software Quality Technician will send a copy of the approved and recorded plan to the PI or TL who originated the plan.

The Individual Software Plan will address the complete life cycle. The content items of the reference life cycle are required and must be addressed, even if the specific reference life cycle names and terminology are not used. Review(s) are specified in the plan within each life cycle phase. The Software Quality Manager when reviewing this plan will consider whether it adequately implements software quality achievement and practices which are applicable to the particular software development effort. It is anticipated that Individual Software Plans will contain life cycles which differ from this reference life cycle but all phases described here must be addressed in any individual plan.

Review: The Individual Software Plan review assures that the planning is complete and consistent. The review also assures that there is sufficient detail to address all phases of development and use.

When the Individual Software Plan is ready for review the developer submits the document to the TAL. The TAL reviews the Individual Software Plan and documents that review. The documentation for the review contains a record of the review comments and their resolution, a plan and timetable for implementation of the resolution, and the personnel responsible for the resolution. The TAL then authorizes the ISP for use.

6.2.3 Life Cycle Phases

Each phase of the life cycle is to be controlled and documented. The following subsections detail the requirements for each phase. It is recognized that there may well be iterations, or revisions in each phase of the life cycle which would then have impacts on the logically subsequent phases of the life cycle. Revisions might be caused by new knowledge of physical processes or end user requirements, or by problems revealed when the earlier phases are implemented in the later phases. Requirements in the following subsection implement these generic requirements (among others):

1. Work during a later phase in the life cycle is always based on a product of an earlier phase, which product is recorded, subject to review, and authorized by the TL for such use.

2. A revision establishing a new recorded and authorized product of a life cycle phase is subject to the same review and authorization steps as the original product.

3. Staff working on a later phase are made aware of the latest authorized product of earlier life cycle phases, and will conform their work and products to this latest authorized product.

4. Upon any authorized revision, work already done in later phases is reviewed as to whether changes in the later phase work are required.

6.2.3.A. REQUIREMENTS PHASE

This phase of software development documents why this software project was initiated and what the software must accomplish. The Individual Software Plan must be completed prior to initiating this phase.

The Software Requirements Specification (SRS) may be developed, used, and reviewed in modules, if the software requirements can be structured in modules. In this case, the top-level modular structure of the SRS, including interconnections, and the specification of one or several modules can be completed first; the following phases can start on these modules.

Requirements modeling and rapid prototyping are among the software development techniques which may be used in developing and confirming the suitability of the requirements. Software developed by rapid prototyping is not the final software product. The use of rapid prototyping must be specified in the Individual Software Plan.

Output: Software Requirements Specification (SRS)

The SRS will explain the following: the purpose, scope, intended users, format and language that is understood by the programming organization and the user, and terminology for the required software product. The SRS includes a general description of the design constraints, functional specifications, attributes, interface(s), and performance requirements. The SRS includes, as appropriate, all the detail covered in Appendix B.1. This document contains enough detailed information for objective verification of the requirements. This document does not prescribe the software design.

Review: The SRS review assures that the requirements are complete, verifiable, consistent, and formatted to provide traceability of requirements throughout the life cycle process. The review also assures that there is sufficient detail available to complete the software design.

When the SRS is ready for review the developer submits the document to the TL (or next management level if the developer is the TL). The TL reviews the SRS and documents that review. The documentation for the review contains a record of the review comments and their resolution, a plan and timetable for implementation of the resolution(s), and the personnel responsible for the resolution. Review documentation becomes a permanent record. The TL then authorizes the SRS for use as a basis of the software design phase.

The TL may authorize the SRS for provisional use as the basis for the design phase before the review is complete. If the review results in changes to the SRS, then work already done in the design phase is reviewed as to whether changes in the design are required by the changes in the SRS.

Phase Completion: The SRS phase is completed when the SRS is submitted by the developer as containing the required content, is ready for review, and the SRS is authorized by the TL (or next management level if the developer is the TL) for use as the basis of the software design phase.

Configuration Management(CM): At phase completion the SRS is submitted to the Software Quality Manager and becomes an element of the Configuration Management system. The SQM assigns a Configuration Item Identifier at this time and the SRS becomes baselined. Any later approved, revised SRS also becomes an element of the Configuration Management system. If a document is approved for provisional use it will be entered into the Configuration Management system by becoming a Configuration Item and will be assigned a Configuration Item Identifier.

6.2.3.B. DESIGN PHASE

This phase of the software development documents the programming strategy that was decided on, describes the major components of the design with the aid of items such as structure charts, flow diagrams, decision table, or pseudocode and relates the design to the SRS thus ensuring traceability. The design documents may be developed, used, and reviewed in modules, if the software design can be structured in modules.

Design models and rapid prototype may be used to develop and confirm the suitability of portions of the design. Software developed by rapid prototyping is not the final software product. The use of rapid prototyping must be specified in the Individual Software Plan.

The design documentation may be a single document called the Software Design Description (SDD) or may be two documents called the Software Preliminary Design (SPD) and the Software Detailed Design Description (SDDD). The end result (the level of detail) of the documentation is the same. The design documentation includes, as appropriate, all the detail covered in Appendix B.2.

Optional Output: Software Preliminary Design (SPD)

Included in this document are items such as the overall description of the input and output data, the flow of information and data, external files and databases, control flow, control logic, data structures, and design limitations. This document expands on the SRS to describe the approach for how the requirements will be met.

Optional Output: Software Detailed Design Description (SDDD)

This document extends the design down to the module level, giving a clear description of the major tasks and processing that occurs within a module. Included is all data input, data output, allowable tolerances for inputs and outputs, interfaces with other modules, and verification activities.

Output: Software Design Description (SDD)

This document contains all the detail mentioned in the two previous documents.

Review: The design documentation review assesses and verifies the technical adequacy of the selected design approach; checks the design compatibility with the functional and performance requirements of the SRS; verifies the existence and compatibility of all interfaces between software, hardware, and user or between database and user; reviews compatibility with the software interface with which the database is required to interact; and assures that the design addresses all the elements in the software requirements specification. This review also assesses the technical risks of the product's design. The complexity of the software design may require the performance of multiple design reviews.

When the design documentation is ready for review the developer submits the design documentation to the TL (or next management level if the developer is the TL). The TL reviews the design documentation and documents that review. The documentation for the review contains a record of the review comments and their resolution, a plan and timetable for implementation of the resolution(s), and the personnel responsible for the resolution. Review documentation becomes a permanent record. The TL then authorizes the design documentation for use as a basis of the programming phase.

The TL may authorize the Software Design Description for provisional use as the basis for the programming phase before the review is complete. If the review results in changes to the design documentation, then work already done in the programming phase is reviewed as to whether changes in the coding are required by the changes in the design documentation. If the review results in changes to the design documentation that in turn need to be reflected in the SRS, the SRS is revised, reviewed and authorized for use.

Phase Completion: The design phase is completed when the design documentation is submitted by the developer as containing the required content, is ready for review, and the design documentation is authorized by the TL (or next management level if the developer is the TL) for use as the basis of the programming phase.

Configuration Management (CM): At phase completion the design documentation is submitted to the Software Quality Manager and becomes an element of the Configuration Management system. The SQM assigns a Configuration Item Identifier at this time and the design documentation becomes baselined. Any later approved, revised design documentation also becomes an element of the Configuration Management system. If a document is approved for provisional use it will be entered into the Configuration Management system by becoming a Configuration Item and will be assigned a Configuration Item Identifier.

6.2.3.C. PROGRAMMING PHASE

This phase of software development translates the design into a computer language. Also during this phase, due to design changes made while coding, modification of the SRS, of the design documentation, and of the design based test cases occur, with the modification subject to the same level of review and authorization for use as the original documentation.

Output: Source Code

This document is a listing (on computer-readable or human-readable media such as microfiche, microfilm, or paper) of the source code and of system commands needed to run the code. The source code shall be appropriately commented.

Review: The review examines the source code for adherence to coding standards and assures that the source code correctly embodies the design as specified by the design documentation and the SRS.

When the source code is ready for review the developer submits the source code to the TL (or next management level if the developer is the TL). The TL reviews the source code and documents that review. The documentation for the review contains a record of the review comments and their resolution, a plan and timetable for implementation of the resolution(s), and the personnel responsible for the resolution. Review documentation becomes a permanent record. The TL then authorizes the source code for use as a basis of the test, debug, and verification phase.

The TL may authorize the Source code for provisional use as the basis for the next phase before the review is complete. If the review results in changes to the source code, then work already done in the test, debug, and verification phase is reviewed as to whether changes in the next phase are required by the changes in the source code. If the review results in changes to the source code that in turn impact the design documentation and the SRS, the design documentation and the SRS are revised, reviewed, and authorized for use.

Phase Completion: The programming phase is completed when the source code is submitted, by the developer, containing the required content, is ready for the review and the source code is authorized by the TL (or next management level if the developer is the TL) for use as the basis of the test, debug, and verification phase.

Configuration Management (CM): At phase completion the source code (on appropriate permanent media) along with the "Software Summary" form (see Appendix E) is submitted to the Software Quality Manager and becomes an element of the Configuration Management system. The SQM assigns a Configuration Item Identifier at this time and the code becomes baselined. Any later approved revised source code becomes an element of the Configuration Management system. If a document or code is approved for provisional use it will be entered into the Configuration Management system by becoming a Configuration Item and will be assigned a Configuration Item Identifier.

6.2.3.D. TEST, DEBUG, AND VERIFICATION PHASE

This phase of software development tests, debugs, and verifies the program (i.e. source code) to determine whether the code accurately performs the requirements described in the SRS. All verification activities will be documented. Verification shall assure that the software performs the intended function and does not perform any degrading or unintended functions. The goal of test, debug, and verification is to develop a set of test cases that have the highest probability of detecting the most errors in order to identify under what conditions the software does not perform properly. This phase can start on a modular basis when the module of source code is complete (submitted for review and authorized by the TL for use in this next phase). A driver of the module for test purposes, or a skeleton of the whole program, may be used as a test driver.

The skeleton should contain the full implementation of the top-level design of the program, and may contain stubs for other modules that are not yet implemented. For stubs modules that interact with the module under test, the stubs should provide data transfer features mimicking those in the final modules. A sufficient subset of the tests shall be repeated when the full source program is completed.

Writing of a Software Test and Verification Plan (STVP), and development of test problems, should begin earlier in parallel with other phases, using and exchanging available information. The Test, Debug, and Verification Phase does not preclude informal and unrecorded testing by a developer or by several developers whose work is related, during the Programming Phase.

The Test, Debug, and Verification Phase really has two subelements: debugging and final verification tests (acceptance tests). The debugging phase overlaps with the programming phase. Progress of debugging during the programming phase may be an appropriate subject for archiving in a large software project, but not required unless called for in the Individual Software Plan or the Software Test and Verification Plan (STVP). Final verification testing includes a sufficient set of tests on the final source code, and review of completed tests on earlier development versions of the source code. Such reviews will consider whether the tests are verifying elements of the code that have not been further modified or impacted by code changes leading to the final source code.

Model validation is addressed in Section 6.8. Model validation is application dependent. Model validation is not complete at this phase as it is an ongoing effort, which continues until licensing.

Software testing and verification documentation include a plan that describes the tasks and criteria for accomplishing the verification of the source code and a report that details the results of the testing and verification activities. The testing and verification documentation may be a single document called the "Software Test and Verification Plan/Report" or may be two documents called the "Software Test and Verification Plan (STVP) " and the "Software Test, Debug, and Verification Report (STDVR)". The end result (the level of detail) of the documentation is the same. The documentation includes, as appropriate, all the detail covered in Appendix B.3 and B.4.

Optional Output: Software Test and Verification Plan (STVP)

This document specifies the hardware and system software configuration(s) for which the software is designed. The plan relates test and verification activities directly to the SRS and design based test cases. In those cases where testing is used to ensure that requirements were met in the software design, the plan provides traceability from requirements to design as implemented in the source code.

Optional Output: Software Test, Debug, and Verification Report (STDVR)

This document reports the results of the execution of the test and verification activities. It reports all tests run that verify that the program functions correctly. It also reports exercises of all possible paths and the use of valid and invalid data during testing. The report also includes the results of all reviews and audits, and contain a summary of the status of the software.

Output: Software Test and Verification Plan/Report

This document contains all the detail mentioned in the two preceding documents. Standard output is the Software Test and Verification Plan/Report that contains the information specified in Appendix B.3 and B.4.

Review: The test and verification documentation review assesses the adequacy of the test planning, the validity of the test cases, and the completeness of the test and verification planning. The review also verifies the accuracy of the results, as stated in the report, and assesses the completeness of the test and verification activity.

When the test, debug, and verification documentation is ready for Technical Review the developer submits the documentation to the TL (or next management level if the developer is the TL). The Technical Review is conducted in accordance with QP 2.4, "Technical Review" or QP 3.3, "Review of Technical Publications". The reviewers assigned will be independent of the performance of the original work. The TL reviews the documentation and documents that review. The documentation for the review contains a record of the review comments and their resolution, a plan and timetable for implementation of the resolution(s), the personnel responsible for the resolution, and a statement regarding the completeness of this verification effort. Review documentation becomes a permanent record. If the review results in changes to the plan or to the planning part of the plan/report, then work already done for the report is reviewed to see whether the changes in the planning will cause changes in the work done for the report. Changes are made in accordance with Section 4.2, "Configuration Change Control".

Phase Completion: The test, debug, and verification phase is completed when the test, debug, and verification documentation is submitted by the developer to the TL as containing the required content, is ready for review, and the documentation is authorized by the TL (or next management level if the developer is the TL) for use.

Configuration Management (CM): At phase completion the test, debug, and verification documentation is submitted to the Software Quality Manager and becomes an element of the Configuration Management system. The SQM assigns a Configuration Item Identifier at this time and the test, debug, and verification documentation becomes baselined. Any later approved revised documentation also becomes an element of the Configuration Management system. If a document or code is approved for provisional use it will be entered into the Configuration Management system by becoming a Configuration Item and will be assigned a Configuration Item Identifier.

6.2.4 Reviews

The documentation produced in accordance with the life cycle plan will be reviewed at times specified by the Individual Software Plan (See Section 6.2.2). These reviews will be conducted by at least one person who is qualified to judge the progress and direction of the programming effort, usually the TL unless the TL has participated in the work being reviewed. The reviews will consider both the technical adequacy of the documentation and its adherence to the Individual Software Plan. Documentation of the results of each review will be given along with the reviewed documentation to the Software Quality Manager for inclusion in the Software Records Management System.

6.2.4.1 Review of Manuals

The TL responsible for the development of the Theoretical and User Manuals or any other manuals reviews and approves the final manual or manuals in accordance with QP 3.3, "Review of Technical Publications". The TL responsible for the development of additional documentation due to the inadequacies of manuals from acquired, existing, or commercial software reviews the new documentation and approves its final form. The review of the additional documentation is documented. The documentation for the review contains a record of the review comments and their resolution, a plan and timetable for implementation of the resolution(s), and the personnel responsible for the resolution. Review documentation becomes a permanent record.

Configuration Management (CM): After the review is completed, the review documents and the manual(s) are submitted to the Software Quality Manager and becomes an element of the Configuration Management system. The SQM assigns a Configuration Item Identifier at this time and the documentation becomes baselined. Any later revised, reviewed, and approved documentation also becomes an element of the Configuration Management system. If a document is approved for provisional use it will be entered into the Configuration Management system by becoming a Configuration Item and will be assigned a Configuration Item Identifier.

6.3 Acquired, Existing, and Commercial Software

This section establishes the criteria for purchasing, configuration management, and test, debug, and verification of acquired, existing, and commercial software. This section establishes criteria for software conversion.

6.3.1 Purchasing

Software deemed appropriate for use at LLNL that was developed outside of LLNL-YMP may be acquired or purchased. Software which is considered large SES requires an approved Individual Software Plan which contains an acquisition plan for the software. Only large SES requires an Individual Software Plan and an acquisition plan; all other acquisitions are controlled under normal purchasing controls.

Requests for software acquisition include appropriate criteria to enable the software received to comply with required sections of this Plan. Those sections not met by the software are completed by the acquirer (user) in the relevant phase of the life cycle that is incomplete or, if that is not possible, the reason is documented and maintained with the software and distributed to all users. The following documentation is required for Acquired or Commercial software:

* The documentation, including User's Manuals, provided by the vendor or supplier and a record of its local physical location.

* A record of the software version number.

* A record of the documentation revision number(s)

The user responsible for purchasing or acquiring software is also responsible for assuring the accuracy of results, identification of software errors, assessing the impact of software errors discovered by other users, or while verifying the software. The problems and errors encountered during application are reported to responsible TL and to the vendor or supplier, as appropriate.

Procurement of software is accomplished in accordance with requirements found in procedure QP 4.0, "Procurement Document Control", and in procedure QP 7.0, "Control of Purchased Materials, Equipment, and Services", as applicable.

6.3.1.1 Acquisition Plan

The Acquisition Plan contains a brief description of the software, its intended use, and the conditions that the software is expected to be in when it is received (including all documentation to be received). The plan also states the software category. The plan describes the requirements for bringing documents into compliance with the required documentation of the Software Quality Assurance Plan. The plan also describes the verification activities to be applied to the software. The plan states what vendor verifications are to be supplied (i.e. benchmarks, test cases and results, integration and/or operational checks) and what additional or confirmatory verification activities are to be performed by LLNL. Finally, if further development is to be conducted at LLNL, the plan contains all necessary requirements for in-house development, when not addressed by an Individual Software Plan. Acquisition plans are reviewed in accordance with Section 6.2.2, "Individual Software Plan (ISP)", subsection "Review".

6.3.2 Existing Software

Existing software is software that was written prior to the approval of the Software Quality Assurance Plan and will be used in level I and II activities. A Software Category Selection form, Appendix C, is required for existing software. For large SES an Individual Software Plan is required. The plan contains a description of the software, its intended use and the existing documentation. The plan describes the method(s) for bringing the documentation into compliance with the requirements of the Software Quality Assurance Plan. The plan also describes the verification activities, confirmatory testing, and additional checks to be applied to the software. If further development is to be conducted at LLNL, the plan contains all the requirements for in-house development.

6.3.3 Configuration Management (CM)

Configuration Management for acquired or existing software at the Configuration Item Baseline level, for large SES software acquired under the controls of this SQAP, occurs at the start of work evaluating the received items. Received documentation, software, and any other items of information, plus the acquisition plan and Software Category Selection form, if applicable to that software type, are submitted to the Software Quality Manager for entry to Configuration Management and recording in the Master Log. Software is entered into the Configuration Management system by submitting the Software Category Selection form (See Appendix C) to the Software Quality Manager. For commercial software other than large SES, this will be done as soon as a decision to use the software has been made and the software is loaded and working on the computer system. A new Software Category form is to be submitted for each new version or release and when the technical contact is changed. Changes made to acquired, existing, or commercial software that are not for compatibility or portability concerns are required to be treated as new software and are under the constraints of new software.

6.3.4 Test, Debug, and Verification

Acquired and existing software and changed acquired and existing software will be tested and debugged by the assigned software developer prior to its release. The test, debug, and verification activity determines whether the software accurately performs the functions and mathematics desired for software developed outside of the life cycle process. The functional bases for developing the tests are the theoretical manual as required by NUREG-0856 and the requirements identified in the acquisition plan. Verification shall assure that the software performs the intended function and does not perform any degrading or unintended functions. Verification involves conditions necessary to exercise the software, identify boundary conditions, and provide a suitable benchmark for installation. If the verification requirements cannot be met, then that portion of the code shall be identified and controlled. The extent to which the software is tested is determined largely by the complexity of the software. A simple code performing algorithms generally accepted as correct can probably be completely and accurately verified by the programmer using ordinary debugging techniques. Examples of these techniques are 1) a line-by-line inspection; 2) testing of individual modules or subroutines of the program; and 3) comparing computed results against hand-calculated numbers. Generally, number 3 is considered the final test of a program accuracy for simple codes. The effort that the programmer goes through to test, debug, and verify the software will be documented along with the results.

For software that includes complex and original modeling approaches, the test and debug approach to verification will generally be insufficient for determining the desired accuracy of the software for the full range of its possible application, because calculating more than a few results by hand to compare the program's output may be too time consuming. For this type of software, further verification is called for by Section 6.7, "Application Verification".

If the software was developed commercially or before the initial issue of this plan, the test, debug, and verification activities that were previously performed must be described. If this information is not available verification as described above will be done. Widespread use of the software may be cited as evidence providing additional support for its accuracy.

Model validation is addressed in Section 6.8. Model validation is application dependent. Model validation is not complete at this phase as it is an on going effort which continues until licensing.

6.3.5 Software Conversion

Software conversion is the task of altering software designed to be used on a computer system and/or peripheral hardware other than that for which it was designed. Conversion includes all modifications and tests made to input/output or the source code or additional software written to run the original software on the new system and verification activities necessary to assure that the software operates properly and does not perform any degrading functions. The conversion process is documented and maintained for the specific version of the software and the computer system on which it is installed. Documentation includes, but is not limited to, the following:

A. Software name and version/release number and the hardware make and model number(s) that the software originally operated on.

B. Software conversion number (if the developer re-identified the software with a LLNL-YMP number)

C. New hardware make and model number(s).

D. Description of what was done. The description contains enough detail so that the process can be repeated and describes the test cases used.

All documentation concerning the conversion process and verification activities are reviewed. This review assures technical adequacy of the documentation and the conversion process. This review is documented. The documentation for the review contains a record of the review comments and their resolution, a plan and timetable for implementation of the resolution, and the personnel responsible for the resolution. All conversion documentation, including review documentation, is a permanent record. Configuration management applies to converted software using the original product as the initial baseline. Configuration management records document the conversion and include any modification, configuration changes, or additional software required to make the software functional. Changes to converted software are evaluated and performed in accordance with Section 4.2.

6.4 Scientific Software Notebook (Notebook)

This section establishes the criteria for the use, configuration management, and review of the Notebook. Notebooks are used for Medium SES and Large Calculational Non-SES but may be used in other areas also. The Notebook is a method for reducing the number of controlled documents but must contain the same types of information as required when individual life cycle documents are used. For Medium SES that will be used externally, the required NUREG-0856 documentation will have to be provided separately (i.e. not in a Notebook). The Notebook is used to record data, information, analysis, and work progress on a daily or as appropriate basis. It is the principal recording document from which work related to an activity can be traced. Notebooks are used in accordance with QP 3.4, "Scientific Notebook". The appropriateness of use of the notebook will be considered by the TAL in review of the Activity Plan.

The Notebook is used by qualified individuals who are using a high degree of professional judgment or trial and error methods, or both, in their work. The extent of documentation in the Notebook is such that another qualified individual can use the Notebook to retrace the investigation and confirm the results or repeat the investigation and achieve the same results without recourse to the original investigator.

The Notebook is intended to be the primary recording document from which work can be traced. The Notebook is securely bound and suitable for photocopying of the contents. Entries comply with legibility and permanency criteria specified in QP 3.4.

A Notebook contains, in addition to the requirements of QP 3.4, the following: the content of the referenced lifecycle (even if other terminology or phases are used) and a list of all program names, their version number and Configuration Item Identifier. The purpose of Notebooks is for review of the work and to provide a road map of how the task was completed.

Notebooks are assigned a unique identification number by Document Control.

Configuration Management is handled within the Notebook. All Configuration Item Identifiers (See Section 4.1) are assigned and recorded in the Notebook. Error resolution and change control is documented in the Notebook.

Notebooks may incorporate SQA Plan required documents by reference. If so, such documents shall be submitted to the Software Records Management System for identifier assignment and for archiving. The notebook references this identifier of the document.

Periodic reviews of work conducted under notebook control are accomplished by the TL in accordance with QP 2.4, "Technical Review". The Individual Software Plan specifies when these reviews are to be conducted. Review of a Notebook is completed in accordance with QP 3.4, "Scientific Notebooks".

6.5 Software Documentation

This section describes the documentation required for each category of software (See Section 6.5.3). Appendix B through G give detailed instructions and provide the forms which support the documentation requirements. Appendix B gives a detailed explanation of the content of these documents. Appendix C contains the Software Category Selection form, the instruction for completing the form, and a brief description of the different software categories. Appendix D contains the Software Documentation Checklist for SES documentation required by NUREG-0856. The checklist will be submitted with the documentation. Items requested that are not applicable should be marked as not applicable (N/A). Appendix E contains the Software Summary form and directions for completing the form. Appendix F contains the Software Quality Checklist and Submittal. Appendix G contains the Statement of Analysis - Specified Software Certification.

Documentation must be submitted for independent technical review before being submitted to the Local Records Center. The review will be performed and documented by the TL. QP 2.4, "Technical Review" is to be used for material that will not be published or released with the software and QP 3.3, "Review of Technical Publications" will be used for documentation that will be published or released with the software.

Documentation is divided into two categories: one is the documentation required by NUREG-0856; and the other is the documentation required by life cycle. All documentation should be written in an amount of detail that would allow a potential reviewer or user with knowledge of the intended application to evaluate or use the software effectively. The Individual Software Plan specifies how the various items of documentation are combined.

6.5.1 NUREG-0856 Documentation

NUREG-0856 documentation consists of material meeting the following categories:

A. Theoretical Manual: What the program does (i.e., its purpose) and How the program performs its purpose (i.e., algorithms, sort routines, etc.).

B. Users Manual: How to use the program. Appendix B.6 may be used as a guide.

C. Source code listings: Specifically, a paper hard copy (option: a copy on microfiche), and a computer-readable copy on magnetic tape or floppy disk.

D. Software Summary (See Appendix E).

E. Code assessment and support (See Appendix B.8): Verification and

validation activities descriptions, results summaries and references.
Summary of any independent reviews of the models and methods used.
Summary description of software maintenance program, differences among software versions, and QA program.

F. Continuing Documentation (See Appendix B.9).

(Appendix B may be used as a guide but for complete requirements refer to NUREG-0856)

6.5.2 Life Cycle Documentation

Life cycle documentation consist of the following:

A. Individual Software Plan (See Section 6.1.2 and 6.2.2)

B. Software Requirements Specification (See Section 6.2.3.A).

C. Design Documentation (See Section 6.2.3.B).

D. Verification Documentation (See Section 6.2.3.D).

E. Software Category Selection (See Section 6.1.1).

F. Software Documentation Checklist (See Section 6.5).

G. Software Quality Checklist and Submittal (See Section 6.5).

H. Statement of Analysis - Specified Software Certification (See Section 6.5).

For a commercial or acquired software, an effort must be made to obtain documentation from the vendor/supplier or to have this documentation produced. The TL assigned to that software must justify the missing documentation or furnish the required (new) documentation. The SRS and Design Documentation for commercial software are expected not to be available.

6.5.3 Categories

6.5.3.1 In-House Large SES

For Large SES, the minimum documentation consists of one manual (or the required information contained in several manuals) that includes all documentation required by NUREG-0856 (See Section 6.5.1) and Life Cycle (See Section 6.5.2).

6.5.3.2 Acquired, Existing and Commercial Large SES

For Acquired, Existing, and Commercial Large SES, the minimum documentation consists of one manual (or the required information contained in several manuals) that includes all documentation required by NUREG-0856 and the following:

1. Acquisition Plan
2. Software Category
3. Available development documentation (e.g. SRS, Design Documentation, etc.)
4. Software Summary
5. Documentation as specified in Appendix F, as appropriate.

6.5.3.3 In-House Medium SES

For medium SES, the minimum documentation consists of one manual (or the required information contained in several manuals) that includes all documentation required by NUREG-0856 (See Section 6.5.1) and Life Cycle (See Section 6.5.2). This may be contained within a Notebook.

6.5.3.4 In-House Large Calculational Non-SES

For in-house large calculational non-SES, the minimum documentation consists of one manual (or the required information contained in several manuals) that includes all documentation required by NUREG-0856 and documentation specified in Appendix F, as appropriate.

6.5.3.5 Acquired, Existing, and Commercial Calculational Non-SES

For Acquired, Existing, and Commercial Calculational Non-SES codes, the minimum documentation consists of:

1. A user's manual, obtained from the vendor or supplier.
2. Source-code listings, on paper and on a computer-readable medium, if obtainable.
3. Documentation specified in Appendix F, as appropriate.

6.5.3.6 In-House Non-Calculational Software

For non-calculational software identified as critical in an activity plan, the minimum documentation consists of the following:

1. Individual Software Plan if specified in the activity plan.
2. Software Category Selection (Appendix C) and Software Quality Checklist and Submittal (Appendix F)
3. Statement of Analysis - Specified Software Certification, unless the Individual Software Plan provides an alternate means of controlling authorization for use.

6.5.3.7 In-House Small SES

For Small SES category of software developed in-house, the minimum documentation consists of one manual (or the required information contained in several manuals) that includes all documentation required by NUREG-0856 (See Section 6.5.1) and Life Cycle (See Section 6.5.2).

6.5.3.8 Acquired, Existing, and Commercial Small SES

For acquired, existing, and commercial Small SES, the minimum documentation consists of one manual (or the required information contained in several manuals) that includes all documentation required by NUREG-0856 and the following:

1. Acquisition Plan
2. Software Category
3. Available development documentation (e.g. SRS, Design Documentation, etc.)
4. Software Summary
5. Documentation as specified in Appendix F, as appropriate.

6.5.3.9 Small Calculational Non-SES and Non-Calculational Software

For Small Calculational Non-SES and Non-Calculational categories of software developed in-house or acquired software (including existing and commercial) no documentation is required unless specified in the activity plan.

6.6 Certification of Software for Analyses

Whenever software is used to perform a QA Level I or II analysis (which includes running the software to perform verification, validation, or a design or performance analysis), the investigator must submit a Statement of Analysis-Specific Software Certification (See Appendix G) for that software for that use. This statement contains:

a. The name, version, and release number of each piece of software to be used in the analysis.

b. An identifying number associated with the analysis, and the Quality Assurance Level.

c. A description, with references, of work already completed for meeting the requirements listed in section 6.7, "Application Verification" and section 6.8, "Validation".

d. Future plans, if any, for performing additional application verification and validation to lend support to the conclusions reached in item "g" below.

e. Tentative plans for efforts to ensure that the results of the analysis will be compared with the results of future application-verification and validation work. The plans must assign responsibility for the comparison. The purpose of these plans is to ensure that future results do not change the conclusions reached in item "g" below.

f. Assurance that sections 4.0 and 6.1 through 6.3 (where applicable) have already been completed.

g. A summary which states the software to be used, in its present state of development and documentation, is appropriate and adequate for the intended analysis. This statement is detailed enough to assure the user that the software adequately meets the objectives of the analysis and allows the user the ability to assess whether the input data and assumptions are valid and traceable. This statement contains, in addition to the evidence cited in item "c" above, the limits, ranges, and tolerances associated with the analysis, any specific quality related items not mentioned previously, and any features of development of the software (e.g. the use of expert judgment) that have helped to ensure its quality.

h. For interim analysis the summary must include a review of the specific analysis to verify that required software options will not call incompleted code segments and that incompleted code segments are adequately isolated from the utilized routines or modules so as to preclude affecting the validity and accuracy of the interim calculation. The status of the results of the interim analysis shall be stamped or otherwise marked to qualify use and limitations of incompleted documentation, validation or verification and clearly stated in the preface of all results (See QP 3.5, "Control of Internal Technical Interfaces").

The form shown in Appendix G, Statement of Analysis–Specific Software Certification, shall be used to submit this information. It must be initialed by the software developer or Software Quality Manager as confirmation that the statement in item "f" is correct. It must also be approved by the Task Leader, TAL, and the TPO (if for a QA Level I effort which has not completed verification), who by their signatures express their agreement with the certification statement signed by the TL. The completed form must be submitted to the Software Quality Manager. The Master Log is updated with this information, and a copy is placed in the appropriate file for this particular analysis.

6.6.1 Criteria for Use in Analysis

Given that all requirements have been met and that the appropriate "Statement of Analysis – Specified Software Certification", Appendix G, has been completed, the user may then use this software for Quality Assurance Level I and II analysis. Software may be used for analysis upon completion of verification activities (Section 6.2.3.D or 6.3.3 and 6.6(3)). Software used in analysis must be reviewed according to this section regardless of its state of development. Validation is an on going and application specific activity.

6.6.2 Application (User) Requirements

Procedures are established for user's to provide reasonable assurance that the software used is appropriate for the intended application and that the results produced by these applications are accurate and can be independently reproduced. Procedures are also established for documenting and reviewing software applications that perform technical calculation. Documentation appropriate for a given application or analysis includes the assumptions or approximations employed to develop the input data, input and output files, the software used, and appropriate user documentation for performing the application or analysis. Review of software applications will assess the adequacy of the documentation, the validity of the assumptions or approximations and the accuracy of the results. Review documentation contains a record of the review comments and their resolution, a plan and timetable for implementation of the resolution, and the personnel responsible for the resolution. The application and attendant documentation including review documentation are subject to Configuration Management and are considered a permanent record.

Application(s) whose verification activities are not complete will state so in their documentation. This statement will contain the limits and ranges for which the verification activity has been successfully completed, states what activities have not been completed and describes the risk involved with using the results of the application. Application/analysis results will be distributed to registered users.

6.7 Application Verification

Application verification is an extension of section 6.3.4, "Test, Debug, and Verification" for Existing, Acquired and Commercial Software in that it ascertains that the program correctly performs the mathematics intended and accurately corresponds to its documentation. However, application verification is generally applied to software of greater complexity whose accuracy for the (application-specific) combination of options and input value ranges cannot be determined to the desired level using ordinary testing and debugging techniques. For the purposes of this plan, application verification is considered to be a particular kind of analysis; it is performed by the in-house end user and allows the user to verify the program for his particular application and its associated range of input.

There are three methods of application verification most commonly used:

1) Comparison of results computed by the program with those calculated by hand or analytically.

2) Use of a "Proof of Correctness". (See The Science of Programming, written by David Gries and published by Springer-Verlag in 1981; ISBN 0-387-90641) This technique employs a form of mathematical induction to prove that every portion of a software product is depicted correctly.

3) Benchmarking the program against other software (See Appendix A for definition of "Benchmarking").

An application verification effort may use one or more of these techniques in establishing the desired accuracy of a code. The optimal approach may vary from application to application.

For acquired, existing, and commercial software, verification functions described in the theoretical manual will be done by means similar to those for in-house SES. Verification of particular combinations of options or particular ranges of inputs, which go beyond the scope of the general verification and are needed for a particular application, can be done during Application Verification.

Verification efforts will be fully documented. The documentation must include the method, actual steps taken, tests run, and the results. These documents must be subjected to a review in accordance with QP 2.4, "Technical Review" or QP 3.3, "Review of Technical Publications". If the verification effort or its review reveals that additional changes are needed to the software, these changes will be handled by the TL of the software. These changes, and indeed all changes are handled in accordance with Section 4.2, "Configuration Change Control" to ensure that the altered software is assigned a new version or release number, any needed additional verification or reverification efforts are performed, and all documentation is kept up to date. The TL for each "Application Verification" exercise will supply to the Software Quality Manager all the documentation resulting from the exercise or an appropriate cross-reference for inclusion in the Software Records Management System. The Software Quality Manager will make certain that any findings brought out through the verification process are distributed among the registered users.

6.8 Validation

Validation is the process of judging that a model is a sufficiently accurate representation of a real system for a specified application. Validation provides the support required in paragraph 60.21(c)(1)(ii)(F) of 10CFR Part 60. The validation activity shall be supported by using an appropriate combination of such methods as field tests, in-situ tests, laboratory tests which are representative of field conditions, monitoring data, and natural analog studies. Validation will meet the requirements of NUREG-0856. Validation may use comparisons of model predictions, using appropriately chosen values and boundary conditions input to the software embodying the model (but not values calibrated on the measurements to be compared against), with verified and traceable data from laboratory experiments, field experiments, in-situ testing, and natural analogues. Specific sets of data used in the validation process are identified, and justification is documented for their use. These comparisons and conclusions will usually be published. Validation may also use previous comparisons contributing to validation of equations, concepts, etc., that are components of the model. The final judgment of adequacy of the model for the intended application and of the information supporting this conclusion, may be done by a peer review panel. The DOE-OCRWM Validation Oversight Committee is developing approaches and guidelines for adequate validation.

The activity of validation is broader than software development, involving as it does the identification or development of laboratory or field data, and evaluation of comparisons. The planning, description, review, and control will be addressed in the scientific investigation planning documents and handled in accordance with LLNL-YMP Quality Procedures. The details of the process of Model Validation is not within the scope of the SQAP (See QP 3.0, "Scientific Investigation Control").

The software development activity will have references to the validation activities and resulting documents. For software that embodies a model, validation documentation will include available referenceable information that indicates the limitations and capabilities of the model – i.e., the conditions for which the model is considered to be valid for the specified application.

Validation activities will be documented in accordance with the Code Assessment section of the NUREG–0856. All information regarding the conditions for which the model is valid will be documented. If the validation effort or its review reveals that additional changes are needed to the model, these changes will be handled by the TL. Once validation is initiated, any software changes must be evaluated through both the verification and validation processes to the degree that the change impacts the model and the code. The model being validated is that which is embodied in the code. Changes to the software are accomplished in accordance with Section 4.2, "Configuration Change Control" to ensure that the modified model and associated software are assigned a new version or release number, any additional needed validation efforts are performed, and all documentation is kept up to date. The Software Quality Manager collects and stores all documentation, software, and references in the software's permanent file and make certain that summary information of the validation process is distributed among all registered users.

6.9 Software Product Completion

This section establishes the criteria for Software Product completion and post-release maintenance. When the requirements in Section 6.1 through 6.8 have been completed for a particular task (activity) the TL responsible for that task (activity) will submit a "Memo of Completion of Software Quality Requirements" (Completion Memo) to the SQM and a "Software Product Summary" to the TAL.

All software quality requirement efforts for a task (activity) must be completed before the results of the tasks (activity) are used in support of an application for a license from the Nuclear Regulatory Commission.

6.9.1 Product Release

The Product Release is verified in accordance with QP 2.4, "Technical Review".

6.9.2 Completion Memo and Software Product Summary

The Completion Memo lists all the requirements, their resolution, and location of documentation. The memo is submitted to the SQM for approval and archiving.

The Software Product Summary lists all items associated with the product and the location of all documentation. After the Summary is approved, the TAL will send the approved summary to the Software Quality Manager for archiving. The Software Quality Technician makes a copy of the approved Summary and forwards it to the developer. The purpose is to provide a map of the software product.

6.9.3 Distribution and Transfer

The Software Quality Technician maintains a Registered Users Distribution Log to record YMP recipients of the LLNL developed software product. The log contains a record of the material transmitted and the baseline identifiers. The Log is updated each time the software product is distributed.

Software being transmitted from one location to another is verified to see that the software was transmitted correctly. Appropriate test cases and their results are transmitted along with software to be used to verify that the software performs correctly.

6.9.4 Records

All records retained by the Software Records Management System are turned over to the Local Records Center in accordance with QP 17.0, "Quality Assurance Records".

6.9.5 Post-Release Maintenance

The TL responsible for the software product is responsible for the post-release maintenance. Post-Release maintenance consists of investigating, documenting, and resolving errors and changes (either in the software, or the associated documentation) that are reported by Registered Users. Error correction is accomplished in accordance with Section 4.2, "Configuration Change Control". The TL also notifies Registered Users that an error has been reported and of the error's resolution in accordance with QP 3.5, "Control of Internal Technical Interface".

Appendix A

Definitions

This Appendix contains the "Definitions" of terms used in this Plan.

APPENDIX A -- DEFINITIONS

Acquired Software: Software obtained from a source outside LLNL-YMP.

Application verification: An advanced phase of verification for software that is so complex that not all combinations of options and input ranges can be verified. When using such software, the end user must verify the software for the particular conditions of the application.

Benchmarking: The comparison of the results of a particular test case run on a software package with the results of the same test case run on a different software package, where both codes were designed to solve comparable problems.

Calculational Non-SES: Software that can contain complex mathematics, but does not involve modeling of scientific or engineering investigation or design applications. Some examples include statistical packages, graphics packages, mesh generators, spread-sheet programs, software operating analytical instruments, and embedded codes.

Commercial software: Software that is not sponsored by LLNL. It is available to the public on a fee or no-fee basis, and may be used outside of the YMP project.

Computer program, computer code, code: Is a set of computer instructions for performing the operations specified in a numerical model.

Configuration item: Elements of the software product placed under Configuration Management are called configuration items, and each is assigned a unique identifier. Items so identified, include software and documentation.

Configuration Management (CM): As used for computer software: (1) A system for orderly control of software, including methods used for labeling, changing, and storing software and its associated documentation. (2) The systematic evaluation, coordination, approval, and implementation of all approved changes in an item of software after establishment of its configuration.

In-house software: Software developed under LLNL-YMP auspices to fulfill a particular LLNL requirement. Includes software developed under contract to LLNL but does not include software developed independent of YMP.

Life cycle: See Software Development Life Cycle

APPENDIX A (continued)

Model: A representation of a physical system. The following hierarchy is used in discussing models in this plan.

1. Physical Process/System is the real world.
2. *Physical Model:* A representation of a physical system, based on scientific principles and laws, that describes the transformation of one set of input information into another set of output information.
3. *Mathematical model:* A mathematical representation of a system or process.
4. *Numerical model:* A representation of a process or system using numerical methods.

Non-calculational software: Software that performs data or symbol manipulation but not mathematical calculations. Examples include compilers, word processors, operating systems, etc.

Principal Investigator (PI): The person with assigned responsibility for conducting the activities described in the Activity Plan, which can include development and use of software.

Registered User: Software user involved in Quality Level I and II activities for YMP that are identified in the Registered User log.

Release: A unique issue of the software product that generally has different capabilities or contains corrections to errors that existed in the previous issue of the software product.

Revision: A unique issue of documentation that generally contains corrections to errors that existed in the previous issue of the documentation.

Scientific and Engineering Software (SES): Software that specifies operations according to a physical or mathematical model or that uses a numerical method and supplies primary data or analysis used in support of Level I and II activities. This category includes scientific, engineering, and mathematical modeling software that use a numerical method. Typical functions for this software include geochemical models, repository or waste package performance assessment, safety and reliability studies, engineering design, etc. Commercial software packages applied to these types of activities are included. This category includes mathematical subroutine libraries used in SES.

Software: A set of computer operations specified in any compiler language that can be translated unambiguously into machine language. (Operations specified in machine language are also software.) This definition includes databases.

APPENDIX A (continued)

Software Quality Manager (SQM): The SQM is assigned by the Deputy Project Leader and is responsible for overseeing implementation of this plan. The SQM's tasks are described in Section 3.3 of this plan.

Software Quality Technician (SQT): The SQT is assigned by the TAL responsible for the project and assists the SQM. The SQT's tasks are described in Section 3.3 of this plan.

Software Development Life Cycle: A method of project planning and documentation for the development of a software product. Life cycle allows optimal traceability regarding the goals, restrictions, decisions made, and current progress of a code for efforts so extensive that the personnel involved may change during the life of the effort.

Technical contact: The technical professional identified on the software summary form, usually the author of or current expert on the software.

Technical review: A documented critical review using QP 2.4, "Technical Review" performed by personnel who have technical expertise at least equivalent to that required for the original work.

Validation: The documented confirmation that the model under review is suitable for an intended purpose. Validation includes assurance that a physical model, as embodied in software, is a sufficiently accurate representation of the intended physical system or process.

Verification: The documented confirmation that the software performs correctly the mathematical and logical operations described in documents pertaining to the software.

Version: A unique implementation of software that generally has different capabilities or contains corrections to errors that existed in the previous implementation of the software.

Appendix B

Requirements for Software Documentation

This Appendix contains the "Requirements for Software Documentation" called out in Section 6.2.3 of this Plan.

APPENDIX B -- REQUIREMENTS FOR SOFTWARE DOCUMENTATION

B.1 Software Requirements Specification (SRS)

The SRS identifies the capabilities the software must possess and the constraints within which it must operate. Each requirement is defined in such a way that its achievement can be verified and and to the extent possible validated. A specific capability of software can be called a requirement only if its achievement can be verified by a prescribed method. The SRS contains the following:

B.1.1 Design Constraints

The SRS defines the design constraints that are imposed on the software by existing standards and regulations, hardware limitations, operational considerations, database and software interaction considerations, or the nature of the software.

B.1.2 Functional Requirements

The SRS defines the output which is to be obtained from the input to the software and the definition, units, limits and tolerances associated with the inputs and outputs. The functional specification of the relationship of the output to the input requires the unambiguous description of concepts and mathematical statement of problems. These statements and concepts describe the model of the physical processes, and the logical and physical relation of the outputs to the inputs. They do not describe solution algorithms, unless the algorithm is needed to specify the concept, or the purpose of the program development is to implement a designated algorithm. Modularity and decomposition into sub-modules are encouraged in developing the functional requirements. Parameters transferred between modules are described by definition, units, limits, and tolerances. The SRS includes the criteria for comparison of the results to the physical system or process that the software represents, and that can be used for validation.

The SRS describes the source of each data item so that the information in the database can be traced. The Level of Quality Assurance of each data item is also specified where applicable. If coefficients or parameters are included in the data, the equations for which the data apply are identified. Uncertainties in the values of numerical data are included, when available. Index files defined for the database are described.

APPENDIX B (continued)

B.1.3 Attributes

The SRS defines the attributes of the software pertaining to non-time-related issued such as portability, efficiency, security, or maintainability.

B.1.4 Interfaces

The SRS defines the interface requirements, if any, imposed on the software to satisfy user, hardware, and communication needs.

B.1.5 Performance Requirements

The SRS define the time-related requirements of the software (i.e. processing speed, recovery time, or throughput) as well as any size-related requirements such as storage requirements; the number of simultaneous user to be supported; number of files and records to be handled; and sizes of records, tables, and files.

B.1.6 Verification and Validation Requirements of Software

The SRS references any generic requirements on verification or validation. Any project-specific requirements on types and degree of validation, methods of verification or validation or comparisons of output results to a physical system or process are stated. The degree of accuracy desired or required, any qualifications or quantitative requirements of the comparisons, are also stated.

APPENDIX B (continued)

B.2 Design Documentation

The design documentation (the Software Preliminary Design (SPD) and Software Detailed Design Description (SDDD) or the Software Design Description (SDD)) determines the architecture of the final software. Its purpose is to separate the software into functional parts so that each part achieves, as independently as possible, the capabilities defined in the SRS. The design documentation provides a functional decomposition of the software into its components, and thus determines its logical structure.The design is described in a manner that is easily traceable to the SRS. The design documentation contains the following:

B.2.a. Exposition of the algorithms and equations of solution methods, and their limits, the logic controlling the solution scheme, and the data operations that are to be performed within the software are included. All functional requirements, as specified in the SRS, are addressed.

B.2.b. Exposition of the structure of the program, in terms of its subroutines or equivalent modules, including: 1) the logic concerning the calling sequence of subroutines; 2) the data elements transferred into or out of each subroutine, whether via the calling statement or via internal and external databases, and 3) the functions of each subroutine. The relation of the functions in subroutines to the functional requirements in the SRS is stated. The data elements in the SDD cover the parameters (input, output, and transferred between major functional modules) described in the SRS and may include additional parameters introduced in the SDD for control or calculational purposes. Attributes of data, expanding beyond the attributes described in the SRS, may include type, word length, array size, where stored in the program's data structure, and source (e.g., input from where, or where created or modified).

B.2.c. A description is provided for the following: the format, content, and allowable tolerances of the input and output files; input and solution checks; output reports; all interfaces; and databases (both internal and external) that are to be used.

B.2.d. Design constraints, such as the type and size of the hardware and the computer language to be used, identified in the SRS are addressed.

B.2.e. The methods for achieving the software attributes and performance requirements as specified in the SRS are described.

APPENDIX B (continued)

B.2.f. The coding standards to be used are specified.

B.2.g. A description of the designed-based test cases is provided and verification activities as specified in the SRS are addressed.

B.2.h. The methods for implementing interfaces between the database and other software or between the database and the user are described.

B.2.i. All functional requirements and design constraints, as specified in the SRS (for a database), are addressed.

B.2.j. The methods for achieving the database attributes and performance requirements, as specified in the SRS, are described.

B.2.k. The methods to be used for verifying the information contained in the database are described.

B.2.l. The methods for implementing the database itself are described.

B.2.m. The methods for gathering data are described.

APPENDIX B (continued)

B.3 Software Test and Verification Plan (STVP)

The Software Test and Verification Plan describes the test and verification approach and methods of performance, specifies how errors will be reported and documented, specifies the level of detail at which test and verification will be carried out, and establishes the degree of rigor to be imposed. The Software Test and Verification Plan is organized in a manner that allows traceability to both the software requirements and the software design. The Software Test and Verification Plan contains the following:

B.3.a. Provisions for the verification or test of separately identifiable components of the software.

B.3.b. Describes methods to be used to verify that the requirements in the SRS are implemented in the design (as expressed in the design documentation), the design expressed in the design documentation is implemented in the software, and that the software produces accurate and stable results relative to the problems to be solved.

B.3.c. Describes the tasks, methods, and criteria for accomplishing test and verification of the software.

B.3.d. Identifies the test documentation that is to be prepares for each module and the system.

B.3.e. A verification matrix, in which the requirements of the SRS are referenced to their corresponding Software Test and Verification Plan section.

B.3.f. Provisions for the verification of separately identifiable data sets.

B.3.g. Describes the tasks, methods, and criteria for accomplishing test and verification of the database.

APPENDIX B (continued)

B.4 **Software Test, Debug, and Verification Report (STDVR)**

The Software Test, Debug, and Verification Report describes the results of the execution of the Software Test and Verification Plan. This includes all reviews, audits, and tests required. The Software Test, Debug, and Verification Report summarizes the status of the software as a result of the execution of the Software Test and Verification Plan. It describes any major deficiencies found; provides the results of the reviews, audits, and test; and recommends whether the software is ready for use.

B.4.1. For verification of a database, all experimental data are either peer reviewed, confirmed through the use of corroborative data, redetermined by confirmatory testing, or collected under an equivalent QA program. Experimental data items that have been previously peer reviewed are accepted without further review.

B.4.2. For verification of the database, the results of comparing entries in the database with the original source data are required.

APPENDIX B (continued)

B.5 Theoretical Manual – Mathematical Models and Numerical Methods

The purpose of the Theoretical Manual is to provide a complete explanation of methods used, including a derivation of and justification for the model along with its capabilities and limitations. Use extensive references to publications and point out new procedures developed for the software.

This documentation should be complete enough to serve as a sole basis for review of the methods used in the code. The Theoretical Manual contains the following (as extracted from NUREG-0856) and shall comply with all the requirements of the NUREG.

1. Overall Description

Describe the purpose of the model. Indicate in general terms the data input and output of the model.

2. Structure

Briefly describe the role of each component model (logically distinct subset of the model). Show the contribution of each to the overall solution of the problem. Use flow charts and block diagrams or equivalent to describe the mathematical solution strategy.

3. Numerical Procedure

Describe the numerical solution strategy and computational sequence. Use flowcharts and block diagrams or equivalent. Give references for the basic numerical procedure. If the method solves a large set of equations, show the structure of the equations and how the coefficients were determined; reference sources of all coefficients. Show the relationship between the numerical strategy and the mathematical strategy (e.g. how boundary conditions are introduced).

4. Component Models

For each component model, provide the following descriptions:

a. Purpose. Describe the purpose and scope of the component model. State the input to and output from the model in general terms and the way the information is processed. State under what circumstances the component model is executed.

APPENDIX B (continued)

b. Assumptions and limitations. Describe the assumptions and limitations of the component model. Include simplifying assumptions about the geometry and behavior of the system. Include the known ranges of validity of the model for all variables. For models based partially or wholly on observed or experimental data (empirical or semi-empirical models), state the range and type of data. State any known uncertainty about the model's validity.

c. Notation. Identify all algebraic variables that represent parameters within the equations being programmed. (This requirement does not apply to temporary storage arrays and temporary variable names. Identify only those variables necessary to allow complete tracing of data through the software.) Give the mathematical symbols used in the fundamental equations, their equivalents in the numerical formulation and the computer variable name.

d. Derivation. Cite the original publication in which the component model appeared and any subsequent references that present modifications that lead up to the present form. Depict the derivation starting with generally accepted principles. Justify each step in the derivation, noting how assumptions and limitations are introduced and how any experimental data were used. State clearly the final mathematical form of the model. If published material contains an adequate derivation, a copy of this may be included instead of a new derivation.

e. Application. Discuss how the component model applies to a an analysis. Point out restrictions on extrapolation of the model or use out of range. Describe any restrictions on the use of the model. Discuss any unusual or extreme conditions that would affect the validity of the model.

f. Numerical method type. Identify any numerical methods used that go beyond simple algebra (e.g., finite-difference method).

g. Derivation of numerical model. Derive the numerical procedure from the mathematical component model, giving references for all numerical methods. State the final form of the numerical model and explain the algorithm. Explain how intermediate results are used.

h. Location. Show where the component model is located within the software.

b. Stability and accuracy. Discuss the stability and accuracy of the numerical model, distinguishing between aspects of stability and accuracy that have been proven mathematically and those that have been observed in practice only.

APPENDIX B (continued)

c. Alternatives. Discuss briefly any alternatives to the component model that were considered and why this one was selected.

5. Performance Evaluation

Discuss the overall performance of the entire model, noting under which conditions the model gives acceptable results. Point out specific component models known to perform poorly under certain circumstances. Give any recommendations to follow when executing the model.

APPENDIX B (continued)

B.6 User's Manual

The user's manual allows the user to understand modeling results, and to install and run the software on the user's computer. This manual, along with hard-copy listings, should be sufficient to instruct a user on how to set up and run problems and resolve any difficulties encountered.

Comments within the code or self-documenting features may be referred to in place of a detailed description in the user's manual if the information is complete enough to enable a new user (with programming experience) to run the software and resolve errors. The User's Manual contains the following (as extracted from NUREG-0856) and shall comply with all the requirements of the NUREG.

1. Program Considerations

   a. Program paths. Describe the purpose of each subroutine. Use flowcharts and block diagrams or equivalent to explain the paths the program can take. Show how the computational sequence and solution strategy described in Section B.5.3 "Numerical Procedure" are related to the program flow.

   b. Program options. Discuss the function of each program option, giving special attention to effects of combinations of options. Relate options to the input values that control them.

   c. Data structures. Discuss how data are stored during computation. Describe the purpose and content of important common blocks and arrays. State the array dimensions and describe the indexing algorithm if dynamic dimensioning is used. This information, along with the hard-copy listing, should be sufficient to allow the user to follow the flow of data through the computational sequence.

   d. Initialization. List any values automatically assigned to important variables, including values of physical significance and parameters that affect program execution. Show where the values are initialized and whether they are default values or fixed.

   e. Restart. Describe any restart capabilities of the code and how they are used.

   f. Error Processing. Describe the origin, likely causes, and corrective action (if any) of all error messages (fatal or nonfatal), error switches, and abnormal stops.

APPENDIX B (continued)

2. Data Files.

a. Content. Outline the general content, purpose, and organization of each data file.

b. Use by program. Describe how and when the files are read and written by the program.

c. Auxiliary processing. Describe any available auxiliary programs that create, modify, or use the files.

3. Input Data

a. General considerations.

(1) Techniques. Describe any special input techniques and requirements such as blank fields, order, or field delineation.

(2) Consecutive cases. If the code is able to retain input data from previous cases, give conditions for retention and reinitialization.

(3) Defaults. Give the general conventions governing default values.

b. Individual input records.

(1) Record identifier. Give the line identifier, if any, for this type of record.

(2) Input variables. State the code variables that will contain data given on this record.

(3) Format. Specify the format of this record, if any.

(4) Need. Specify for each variables whether input is necessary or optional for both start and restart runs.

(5) Repetition. State how many of these input records are required.

(6) Units. For each field, state the dimensional units.

(7) Default. State the default value for each field, if any.

(8) Description. Define each variable and discuss its use within the code. State how to assign values in setting up a run.

(9) Range. State the limits for each variable.

APPENDIX B (continued)

4. System Interface

a. System-dependent features. List the external references in the program that must be supplied by the system, and state the purpose of each. Include plot and mathematical libraries, utility programs, and statistical packages, identifying the manufacturer and version used. Omit any intrinsic functions which are standard to the compiler being used.

b. Compiler requirements. Identify all compilers used and any special load or compiler options that are necessary – e.g., large-core-memory addressing.

c. Hardware requirements. Describe all hardware features needed to execute the code and the amount of memory required for a typical case, along with a general rule for determining the necessary memory for varied cases.

d. Command files. Describe the command files necessary to control program initiation, manipulation of files, and interaction with other programs. Give detail appropriate to the degree to which command files contain logic affecting program flow, manipulation of files, and communication among programs. Give examples.

5. Output

Discuss the code output and relate edited output to input options. State the origin and meaning of the output variables. Describe any normalization of results and list associated dimensional units. Describe any graphical capabilities of the code.

6. Sample problems

Include sample problems which demonstrate how the software is used. These problems need not have known solutions or experimental data, but they should exercise a large portion of the available programmed options. These sample problems should use only a reasonable amount of computer time. Input listings and sample output should be given. Discussion on options selection and corresponding results should be included.

APPENDIX B (continued)

B.7 Source-Code Listings

Listings of the source code should be submitted on each of the following media:

1. Print on 8-1/2 by 11" paper
2. Microfiche
3. A computer-readable media, such as magnetic tape or floppy disk

Include these copies for every version and release of the program that is controlled by the Configuration Management system.

B.8 Code Assessment and Support Documentation

This collection of documentation is intended for review and licensing purposes, per NUREG-0856. This document(s) has the purpose of describing all work which sheds light on the adequacy of the software. The goal of the document for licensing purposes is to ensure that the software has been extensively reviewed, verified, and validated. In addition, the document describes steps the applicant is taking to ensure that software performance will not be degraded by future changes. Existing life cycle documentation may be used where it fills the need. The topics in this category called for by NUREG-0856 are:

1. Model review: Describe any projects for independent review of the methods used in the model. Include past and ongoing programs as well as planned ones. Summarize the results of past reviews. Describe any plans for modification of the model as a result of the reviews. Give references for publications.

2. Verification and validation: Describe the program for verification and validation of the software. Include past, ongoing, and planned future activities. Give descriptions, input files, and results for specific tests. The input files should be appropriate for the software version delivered to NRC. State what aspects of the software each test demonstrates and discuss how well the software performed. Describe any plans for modification of the model as a result of the tests. It is acceptable to provide reproductions of publications, output, and reports documenting the tests. The above should be given for two classes of tests:

   a. Tests by the software developer; and

   b. Independent assessment.

APPENDIX B (continued)

3. Maintenance and Quality Assurance: Often there are several versions of software, each with different modifications, capabilities, and limitations. For licensing purposes, it is not necessary to completely verify and validate each version as though it were a separate code. However, it is necessary to ensure that each version is as correct as possible and that an orderly procedure exists for keeping track of the differences between versions. Since this procedure relates directly to the adequacy of the software for licensing purposes, provide a description of the maintenance and quality assurance programs for the software. Include a brief chronology of the software versions.

B.9 Continuing Documentation

Continuing documentation is required by the NRC for their use of the software. Software generally continues to evolve even after a standard version has been released. This evolution includes the development of new capabilities, the detection and repair of errors, and application-oriented modifications. In order to ensure that licensing decisions are made on the basis of up-to-date information, it is necessary that users be kept informed of changes in the software.

Continuing documentation includes the preparation of new material and revision of the existing documentation and error reporting. It also includes computer-readable and paper listings of the current version and new versions as they are released.

1. Updated software summaries. Submit a new software summary form when the information it contains changes, when new software versions are released, or when the Technical Contact changes.

2. Technical contact. Identify a person whom the user can contact directly with technical questions about installing and running the software. The user should be informed whenever a new person is given this responsibility.

3. Documentation revisions. Revise the documentation sent to the user as needed when changes are made in the software, when errors are found in the existing documentation, when new limitations of the model are found, or when new results of the assessment program described in Appendix B.8 appear.

4. Error reporting. When errors or omissions that could affect the validity or appropriateness of the model itself or specific instances of its use are found, these errors must be reported to the user promptly. This includes input errors. Report action taken to correct the errors. State the significance of the error in past, current, and future modeling activities.

APPENDIX B (continued)

5. Computer files. Send the user computer files containing the current software version, new version as they are released, and necessary updates as they are determined. Include the input files for the sample problems (See Appendix B.6(6)). Include all necessary library routines. The means of transmittal may be any reasonably standard medium. The information should be in a standard format readable by a variety of computer systems. All files should be accompanied by a printout from the runs which created them.

6. Paper listings. Provide printed listings of the current version, new versions as they are released, and updates.

7. Description of updates and new versions. All updates and new versions delivered to the user should be accompanied by descriptions of the changes.

8. Response to user questions. Provide responses to questions by the user concerning the model, its use, or installation. Questions will state whether written response, response by telephone, or a meeting is required.

Appendix C

Software Category

Selection

This Appendix contains the "Software Category Selection" form and an explanation on how to complete the form. A new form is to be submitted for each new version or release.

SOFTWARE CATEGORY SELECTION

1. *PI or TL:* _____ 2. *Date:* _____

3. *Category:*
   a. SES [ ] b. Calculational Non-SES [ ] c. Non-Calculational [ ]
   d. Large [ ] e. Medium [ ] f. Small [ ]

4. Type: In-house [ ] Acquired [ ] Commercial [ ] Existing [ ]
[If "Type" is Acquired or Commercial – fill in item #8]

5. *Justification for category and type selected:*

6. Program Name(s) and version/release number:

7. Configuration Item Identifier and Software Quality Manager/Configuration Management initial

8. Vendor or supplier:
   a. Name:
   b. Address:
   c. Telephone:
   d. Technical Contact:

9. Remarks:

10. Approved By: Date:
   a. Title:

SOFTWARE CATEGORY SELECTION (Continued)

Software Category Definitions

Calculational non-SES: Software that can contain complex mathematics, but does not involve modeling or the use of numerical techniques. Some examples include statistical packages, graphics packages, spread-sheet programs, software operating analytical instruments, and embedded codes.

Commercial software: Software that is not sponsored by LLNL-YMP. It is available to the public on a fee or no-fee basis, and may be used outside of the YMP project.

In-house software: Software developed under LLNL-YMP auspices to fulfill a particular LLNL-YMP requirement.

Non-calculational software: Software that performs data or symbol manipulation but not mathematical calculations. Examples include compilers, word processors, operating systems, etc.

Scientific and Engineering Software (SES): Software that specifies operations according to a physical or mathematical model or that uses a numerical method. This category includes scientific, engineering, and mathematical modeling software that use a numerical method. Typical functions for this software include geochemical models, repository or waste package performance assessment, safety and reliability studies, engineering design, etc. Commercial software packages applied to these types of activities are included.

Instructions for "Software Category Selection" form

1. Name of Principal Investigator or Technical Leader, as appropriate.

2. Date form is initiated

3. Check the appropriate box for software category. Refer to section 6.1.1.

4. Check the appropriate box for software type. Refer to section 6.1.1. If the Type" of software is "Acquired" or "Commercial" fill in item #8, with the appropriate information.

5. State the reason for the category and type selected. This may be very brief or may continue under item #9.

6. List the names and version or release numbers of all programs. Use item #9 if more space is needed.

7. This item is completed by the Software Quality Manager.

8. This item is used in conjunction with item #4. The name, address, telephone #, and Technical Contact of the software originator, vendor, or supplier is to be entered here.

9. Additional information for a specific item or any comments may be entered here.

Appendix D

Documentation Checklist

This Appendix contains the "Document Checklist" form. A new form is to be submitted for each new version or release.

Documentation Checklist

SES Documentation required by NUREG-0856

Software Name:_____ Date:_____

DOCUMENTATION	LOCATION
A. **Theoretical Manual** - Mathematical Models	
1. *Overall Description*	
2. *Structure*	
3. *Numerical procedure*	
4. *Component models*	
a. Purpose	
b. Assumptions and limitations	
c. Notation	
d. Derivation	
e. Application	
f. Numerical method type	
g. Derivation of numerical model	
h. Location	
i. Stability and accuracy	
j. Alternatives	
5. *Performance experience*	
B. **User's Manual**	
1. *Program consideration*	
a. Program Options	
b. Program paths	
c. Data structures	
d. Initialization	
e. Restart	
f. Error processing	
2. *Data files*	
a. Content	
b. Use by program	

Documentation Checklist (Continued)

Software Name:_____ Date:_____

DOCUMENTATION	LOCATION
c. Auxiliary processing	
3. *Input data*	
a. General consideration	
(1) Techniques	
(2) Consecutive cases	
(3) Defaults	
b. Individual input records	
(1) Record Identifier	
(2) Input variables	
(3) Format	
(4) Need	
(5) Repetition	
(6) Units	
(7) Default	
(8) Description	
(9) Range	
4. *System interface*	
a. System dependent features	
b. Compiler requirements	
c. Hardware requirements	
d. Command files	
c. Auxiliary processing	
5. *Output*	
6. *Sample problems*	
E. Software Test and Verification Plan	
F. Software Test, Debug, and Verification Report	
G. Source code listings	
H. Software Quality Checklist and Submittal	
I. Software Summary	
J. Other Code Assessment and Support Documentation	

Appendix E

LLNL YMP
Software Summary

This Appendix contains the "Software Summary" form and an explanation on how to complete the form. A new form is to be submitted for each new version or release and when the responsible software TL is changed.

FEDERAL INFORMATION PROCESSING STANDARD SOFTWARE SUMMARY

01. Summary Date	02. Summary Prepared by (*Name and Phone*)	03. Summary Action
Yr. Mo. Day		New ☐ Replacement ☐ Deletion ☐
04. Software Date	05. Software Title	Previous Internal Software ID
Yr. Mo. Day		07. Internal Software ID
06. Short Title		

08. Software Type	09. Processing Mode	10. Application Area General		Specific
☐ Automated Data System	☐ Interactive	☐ Computer Systems Support/Utility	☐ Management/ Business	
☐ Computer Program	☐ Batch	☐ Scientific/Engineering	☐ Process Control	
☐ Subroutine/Module	☐ Combination	☐ Bibliographic/Textual	☐ Other	

11. Submitting Organization and Address	12. Technical contact(s) and phone

13. Narrative

14. Keywords

15. Computer manufacturer and model	16. Computer operating system	17. Programming language(s)	18. Number of source program statements
19. Computer memory requirements	20. Tape drives	21. Disk requirements	22. Terminals

23. Other operational requirements

24. Software availability	25. Documentation availability
Available ☐ Limited ☐ In-house only ☐	Available ☐ Limited ☐ In-house only ☐

26. FOR SUBMITTING ORGANIZATION USE

Software Summary – Instructions

01. *Summary Date:* Enter date summary prepared. Use Year, Month, Day – format YYMMDD.
02. *Summary Prepared By:* Enter name and phone number (including area code) of individual who prepared this summary.
03. *Summary Action:* Mark the appropriate box for new summary, replacement summary, or deletion of summary. If this software summary is a replacement, enter under "Previous Internal Software ID" the internal software identification as reported in *item 07* of the original summary, and enter the new internal software identification in *item 07* of this form; complete all other items as for a new summary. If a software summary is to be deleted, enter under "Previous Internal Software ID" the internal software identification as reported in *item 07* of the original summary; complete only *items 01, 02, 03,* and *11* on this form.
04. *Software Date:* Enter date software was completed or last updated. Use Year, Month, Day – format YYMMDD.
05. *Software Title:* Make title as descriptive as possible.
06. *Short Title:* (optional) Enter commonly used abbreviation or acronym which identifies the software.
07. *Internal Software ID:* Enter a unique identification number or code.
08. *Software Type:* Mark the appropriate box for an Automated Data System (set of computer programs), Computer Program, or Subroutine/Module, whichever best describes the software.
09. *Processing Mode:* Mark the appropriate box for an Interactive, Batch, or Combination mode, whichever best describes the software.
10. *Application Area:* General: Mark the appropriate box which best describes the general area of application from among the following:Computer System Support/Utility, Process Control, Management/Business, Scientific/Engineering, Bibliographic/Textual, Other. Specific: Specify the sub-area of application: e.g.: "COBOL optimizer" if the general area is "Computer System Support/Utility"; "Payroll" if the general area is Management/Business"; etc. Elaborate here if the general area is "Other".
11. *Submitting Organization and Address:* Identify the organization responsible for the software as completely as possible, to the Branch or Division level, but including Agency Department (Bureau/Administration), Service, Corporation, Commission, or Council. Fill in complete mailing address, including mail code, street address, city, state, and ZIP code.

Technical Contract(s) and Phone: (Usually the Technical Leader responsible for the product) Enter person(s) or office(s) to be contacted for technical information on subject matter and/or operational aspects of software. Include telephone area code. Provide organization name and mailing address, if different from that in *item 11*.

13. *Narrative:* Describe concisely the problem addressed and methods of solution. Include significant factors such as special operating system modifications, security concerns,relationships to other software, input and output media, virtual memory requirements,and unique hardware features. Cite references, if appropriate.
14. *Keywords:* List significant words or phrases which reflect the functions, applications and features of the software. Separate entries with semicolons.
15. *Computer Manufacturer and Model:* Identify mainframe computer(s) on which software is operational.
16. *Computer Operating System:* Enter name, number, and release under which software is operating. Identify enhancements in the *Narrative* (*item 13*).
17. *Programming Language(s):* Identify the language(s) in which the software in written, including version; e.g., ANSI COBOL, FORTRAN V, SIMSCRIPT 11.5, SLEUTH 11, etc.
18. *Number of Source Program Statements:* Include statements in this software, separate macros, called subroutines, etc.
19. *Computer Memory Requirements:* Enter minimum internal memory necessary to execute software, exclusive of memory required for the operating system. Specify words, bytes, characters, etc., and number of bits per unit. Identify virtual memory requirements in the *Narrative* (*item 13*).
20. *Tape Drives:* Identify number needed to operate software. Specify, if critical, manufacturer, model, tracks, recording density, etc.
21. *Disk/Drum Units:* Identify number and size (in same units as "*Memory*" – *item 19*) needed to operate software. Specify, if critical, manufacturer, model, etc.
22. *Terminals:* Identify number of terminals required. Specify, if critical, type, speed, character set, screen/line size, etc.
23. *Other Operational Requirements:* Identify peripheral devices, support software, or related equipment not indicated above, e.g., optical character devices, facsimile, computer-output microfilm, graphic plotters, etc.
24. *Software Availability:* Mark the appropriate box which best describes the software availability from among the following: Available to the Public; Limited Availability (e.g., for government use only); For In-house Use Only. If the software is "Available", include a mail or phone contact point, as well as the price and form in which the software is available, if possible.
25. *Documentation Availability:* Mark the appropriate box which best describes the documentation availability from among the following:Available to the Public; Inadequate for Distribution; For In-house Use Only. If documentation is "Available", include a mail or phone contact point, as well as the price and form in which the documentation is available, if possible. If documentation is presently "Inadequate", show the expected availability date.

For Submitting Organization Use: This area is provided for the use of the organization submitting this summary. It may contain any information deemed useful for internal operation.

Appendix F

Software Quality Checklist and Submittal

This Appendix contains the "Software Quality Checklist and Submittal" forms. A copy of this form accompanies the documentation (including review documentation) submitted to the Software Quality Manager for storage in the Software Records Management System. A new form is to be submitted for each new version or release.

SOFTWARE QUALITY CHECKLIST AND SUBMITTAL

- SES IN-HOUSE -

Name: Phone:
Org: Date:

Product:
VERSION OR RELEASE:
ANALYSIS EFFORT (if applicable):

[*CHECK THE BOXES THAT DESCRIBE THE MATERIAL SUBMITTED WITH THIS FORM*]

1. [ ] Software Category Selection, approved
2. [ ] ISP, approved
3. [ ] Software Requirements Specifications
   [ ] Review
4. Design Documentation, approved
   [ ] Preliminary Design
   [ ] Review
   [ ] Detailed Design
   [ ] Review
5. [ ] Test, Debug, and Verification
   [ ] Review
6. [ ] LLNL YMP Software Summary
7. Software Documentation (include a copy of Appendix D):
   [ ] Theoretical Manual – Mathematical Models & Numerical Methods
   [ ] Review
   [ ] User's Manual
   [ ] Review
   [ ] Source code listing
   [ ] Code assessment and support documentation
8. [ ] Application Verification
   [ ] Certification
   [ ] Review
9. [ ] Validation
   [ ] Certification
   [ ] Review
10. [ ] Analysis
    [ ] Certification
    [ ] Review
11. [ ] Software Documentation Checklist
12. [ ] Software Product Summary
13 [ ] Memo of Completion of Software Quality Requirements, approved

SOFTWARE QUALITY CHECKLIST AND SUBMITTAL
- SES ACQUIRED, EXISTING, AND COMMERCIAL -

Name: Phone:
Org: Date:

Product:
VERSION OR RELEASE:
ANALYSIS EFFORT (if applicable):

[*CHECK THE BOXES THAT DESCRIBE THE MATERIAL SUBMITTED WITH THIS FORM*]

1. [ ] Acquisition Plan, approved.

2. [ ] Software Category Selection, approved

3. [ ] Existing or Acquired Software
   - Test, Debug, and Verification - (not as part of Life Cycle)
     [ ] Review

4. [ ] LLNL YMP Software Summary

5. Software Documentation (include a copy of Appendix D):
   [ ] Theoretical Manual - Mathematical Models & Numerical Methods
     [ ] Review
   [ ] User's Manual
     [ ] Review
   [ ] Source code listing
   [ ] Code assessment and support documentation

6. [ ] Software Conversion documentation, if applicable

7. [ ] Application Verification
     [ ] Certification
     [ ] Review

8. [ ] Validation
     [ ] Certification
     [ ] Review

9. [ ] Analysis
     [ ] Certification
     [ ] Review

10. [ ] Software Product Summary

11. [ ] Memo of Completion of Software Quality Requirements, approved

SOFTWARE QUALITY CHECKLIST AND SUBMITTAL
- CALCULATIONAL NON SES IN-HOUSE -

Name: Phone:
Org: Date:

Product:

VERSION OR RELEASE:
ANALYSIS EFFORT (if applicable):

[*CHECK THE BOXES THAT DESCRIBE THE MATERIAL SUBMITTED WITH THIS FORM*]

1. [ ] Software Category Selection, approved

2. [ ] ISP, approved

3. [ ] Software Requirements Specifications
   [ ] Review

4. [ ] Design Documentation, approved
   [ ] Preliminary Design
   [ ] Review
   [ ] Detailed Design
   [ ] Review

5. [ ] Test, Debug, and Verification
   [ ] Review

6. [ ] Test, Debug, and Verification (not as part of Life Cycle)
   [ ] Review

7. [ ] LLNL YMP Software Summary

8. Software Documentation (include a copy of Appendix D)
   [ ] User's Manual
   [ ] Review
   [ ] Source-code listing

9. [ ] Analysis
   [ ] Certification
   [ ] Review

10. [ ] Software Product Summary

11. [ ] Memo of Completion of Software Quality Requirements, approved

SOFTWARE QUALITY CHECKLIST AND SUBMITTAL
- CALCULATIONAL NON-SES -
ACQUIRED, EXISTING, AND COMMERCIAL

Name: Phone:
Org: Date:

Product:

VERSION OR RELEASE:
ANALYSIS EFFORT (if applicable):

[*CHECK THE BOXES THAT DESCRIBE THE MATERIAL SUBMITTED WITH THIS FORM*]

1. [ ] Software Category Selection, approved

2. [ ] LLNL YMP Software Summary

3. Software Documentation
   - [ ] User's Manual
     - [ ] Review
   - [ ] Source-code listing

4. [ ] Software Conversion documentation, if applicable

5. [ ] Analysis
   - [ ] Certification
   - [ ] Review

6. [ ] Software Product Summary

7. [ ] Memo of Completion of Software Quality Requirements, approved

Appendix G

Statement of Analysis

– Specified Software Certification

This Appendix contains the "Software Quality Checklist and Submittal" forms. A new form is to be submitted for each new version or release.

STATEMENT OF ANALYSIS-SPECIFIED SOFTWARE CERTIFICATION

Submitter: Date:
Org: Phone:
Program name: Version/Release: ISP:
Task: QA Level:
Activity name and number:

Describe the Analysis:

This use if for: [ ] Application verification [ ] Validation [ ] Analysis

---

I CERTIFY THAT THE ABOVE IDENTIFIED CODE IS APPROPRIATE FOR THE INTENDED USE. THE EVIDENCE FOR THIS CERTIFICATION IS THE INFORMATION PROVIDED IN THIS FORM AND THE ACCOMPANYING SUMMARY STATEMENT.

Principal investigator's signature: Date:

Status of Lifecycle:

Software Quality Coordinator's initials: Date:

STATEMENT OF ANALYSIS-SPECIFIED SOFTWARE CERTIFICATION (Continued)

INCLUDE ADDITIONAL SHEETS IF NECESSARY

Application verification work already completed:

Validation work already completed:

Future plans, if any, for additional application verification or validation:

Tentative plans for efforts to ensure that the results of the analysis will be compared with the results of future application verification and validation work:

Approvals:

Task Leader:	Date:
Technical Area Leader:	Date:
TPO (QA Level I only):	Date: