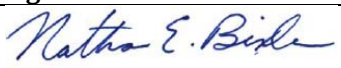
MELCOR Accident Consequence Code System (MACCS)
Software Quality Assurance Plan

Version 1.6

Approval:

	Signature	Date
Project Manager		**9/22/2016**

1 REVISION HISTORY 3
2 PURPOSE 4
2.1 PURPOSE OF THIS DOCUMENT 4
3 REFERENCE DOCUMENTS 4
4 SOFTWARE DEVELOPMENT 4
4.1 COMPUTER CODE REQUIREMENTS 4
4.2 RESPONSIBILITIES 5
4.2.1 Principle Investigator (PI) 5
4.2.2 Program Developer 5
4.2.3 Configuration Management Administrator 5
4.2.4 Others as Assigned 6
5 COMMUNICATIONS 6
5.1 COMMUNICATIONS 6

5.2 TASKS/SCHEDULE 6
6 DOCUMENTATION 6
6.1 QA TRAINING 8
6.2 SOFTWARE REQUIREMENTS SPECIFICATION 8
6.3 IMPLEMENTATION DOCUMENT 8
6.4 TEST PLANS 8
6.5 SOURCE CODE 9
6.6 SAMPLE FILES 9
6.7 USER'S AND MODELS MANUAL 9
6.8 TEST REPORTS 9
6.9 BUG TRACKING DATABASE 10
6.10 CLIENT TRACKING DATABASE 10
6.11 DISTRIBUTION PACKAGE 10
7 REVIEWS AND MEETINGS 10
8 TESTING 10
8.1 UNIT TESTS 11
8.2 INTERNAL RELEASE TESTS 11
8.3 ALPHA RELEASETESTS 11
8.4 BETA RELEASE TESTS 11
8.5 REPORTS AND CORRECTIVE ACTION 11
8.5.1 Reporting a bug 11
8.5.2 Problem Review 12
8.5.3 Investigation of Problems 12
8.5.4 Problem Closure 12
9 TOOLS 12
9.1 TOOLS 12
10 CONFIGURATION MANAGEMENT 13
10.1 SERVER 13
10.2 VERSION NUMBERS 14

1 Revision History

0.1	Draft by Sharon Shannon	05/20/2003
1.0	Signed by Nathan Bixler	06/26/2003
1.1	Modified by Katherine McFadden	08/10/2007
1.2	Modified by Katherine McFadden	09/25/2009
1.3	Modified by Katherine McFadden	06/29/2010
1.4	Modified by Katherine McFadden	10/17/2012
1.5	Modified by Nathan Bixler	04/17/2013
1.6	Modified by Cathy Ottinger Farnum	09/21/2016

2 Purpose

2.1 Purpose of This Document

This document describes the software quality assurance (QA) plan that is being implemented as part of the development and maintenance of the MACCS software suite. The MACCS software suite currently consists of MACCS, COMIDA2, PopMod, WinMACCS, CombineSource, GenHysplit, HyGridConvert, SecPop, LHS, MelMACCS, and REAcct software components.

It should be noted that this project begins with the existing MACCS2 1.12 software, and that no work is planned to back-validate or back-QA the models currently in that version of the software; however, this QA plan does cover existing (or future) MACCS errors and the effort required to resolve these errors. MACCS2 was renamed MACCS in version 3.8.0.2, that was created June 16, 2014. MACCS version 3.9.0, created July 17, 2014, was the first version released that used the new naming.

Topics covered include software development, communications, documentation, reviews, testing, tools used for the project, and configuration management. Artifacts are generated at each phase of the quality process. These artifacts are used to assure the software satisfies its intended mission, is usable, and is maintainable. This plan covers the entire life cycle of each piece of software from the development through the maintenance of the production code. Changes to this plan are to be controlled in the configuration management process.

3 Reference Documents

"Quality Assurance (QA) Plan for Computer Software Supporting the U.S. Nuclear Regulatory Commission's High-Level Waste Management Program," NUREG/CR-4369

This document specifies the overarching requirements and philosophy of the QA plan for the MACCS code suite.

4 Software Development

Software development follows standard engineering and programming practices. Sufficient information should be maintained to allow transfer of the software development to other knowledgeable program developers in the event that the original developers become unavailable.

4.1 Computer Code Requirements

The following internal documentation is to be provided in the main program driver module for each software component:

- program name and version identifier;
- brief description of the program;

- original source of the program;
- name of author;
- history of modification, including name of modifier, extent, and date of modification;

Program output must contain version information and both the time and date of the program execution.

4.2 Responsibilities

4.2.1 Principle Investigator (PI)

Responsibilities include:

- Serves as primary point-of-contact with the NRC.
- Assigns tasks to the MACCS team.
- Tracks the budget.

4.2.2 Program Developers

Responsibilities include:

- Make changes to code as specified in the requirements documents and as directed by the PI.
- Document the design, implementation, and informal testing of code enhancements, modifications and additions adhering to quality assurance documentation requirements.
- Develop formal test plans as assigned by the PI.
- Keep the Configuration Management System up to date with current code enhancements/modifications/additions.
- Write a description of how to use new features as these are implemented. Save these in the Configuration Management System as guided by the quality assurance requirement documents.
- Adhere to requirements outlined in Section 4.1.
- Keep design documents current with implemented model deliveries.
- Document software changes internally and/or externally as needed.

4.2.3 Configuration Management Administrator

- Administrates the configuration management software.
- Backs up the configuration management database and bug tracking database.

4.2.4 Quality Assurance Administrator

- Maintains this Quality Assurance Plan.
- Maintains the QA Training document.
- Periodically reviews these documents as needed, clarifying tasks to other team members. Unresolved issues should be brought to the attention of the PI.

4.2.5 Others, as Assigned

- Enter data into the bug-tracking database.
- Develop models.
- Create requirements documents.
- Implement test plans.
- Review documents as needed.
- Prepare a project plan with estimates of time and costs of tasks.

5 Communications

5.1 Communications

Phone and email communications with the customer should be made as needed by either the PI or the manager. The NRC generally prefers only one point of contact for a project (or two, including the manager). Therefore, if any other member of the team has something to communicate to the customer, he/she should task the PI or manager to do so. Exceptions are when the PI is unavailable for an extended period of time and critical issues need to be discussed.

5.2 Tasks/Schedule

Task assignments are to be agreed upon by team members and the PI. Due dates for each task and action item are to be agreed upon by the team members and the PI. If any team member anticipates that the task or action item cannot be completed by the due date, it is the team member's responsibility to notify the PI as soon as possible. The team member shall not wait until the due date, and then notify the PI that the task is late. The objective is to anticipate when tasks will be late, allowing the PI the maximum flexibility to adjust for the late task.

6 Documentation

The documents in the following table are created and/or maintained in the Configuration Management System (unless otherwise indicated).

Document	Description
QA training	Details of QA and documentation instructions. Includes this document, the QA Training document, and a QA Training Completion document indicating that individual team members have completed the training.
Software Requirements Document	Describes at a high level what the software must do. This exists in template form and is filled out when the software is to be updated.

Implementation Document	Implementation details, suggested test plan and acceptance criteria.
Test Plans	Describes the means to test the product against the model specification. Suggested testing requirements are outlined in the Implementation Document. The testing that was conducted is documented in the Testing Document.
Source Code	Software source code. The primary location of source code is in the Configuration Management System. An additional backup external to the Configuration Management System is maintained on a server.
Sample Files	Standard sample files distributed with the program. These are maintained external to the Configuration Management System on a server.
Data Files	Includes dose conversion factors, census data, county economic data, food ingestion data, site files containing site latitudes and longitudes, and radionuclide decay chain files.
User's and Models Manual	User's and Model Manual refers to any documentation developed for the user (manuals, readme files, etc.). Describes how a user configures and operates the software. Existing manuals are added to the system as they become available. As new models are added, the documentation regarding understanding and using those models should be updated. These are maintained in the Configuration Management System.
Test Results	Results of system level testing and/or unit testing. Can be used as a baseline for possible future regression tests. Test problems are stored on a server.
Bug Tracking Database	A database is used to document suggested improvements and bugs. This is maintained external to the Configuration Management System on a server and is available to selected users in addition to the development team.
Client Data	Data files that list all of the current users and contact information.
Distribution Package	A package of files distributed outside of the Sandia development group. This is stored external to the Configuration Management System on a server.

6.1 QA training

Instructions regarding the proper flow of tasks and documentation from requirements to final testing and release are documented in a document stored in the Configuration Management System called QA Training.docx.

The QA Training document contains the detailed information needed to implement the quality assurance documentation. Each team member needs to read this document. After completion, the team member signs a form titled QA Training Completion username.docx, where username is the users login ID. This documents the team members training history. This document contains the training completion date and document version of that training. When needed, as directed by QA Administrator or PI, the training needs to be refreshed. After completion, the team member documents this on their training completion form.

6.2 Software Requirements Specification

For each version of a code undergoing development, the Requirements form contains a high level description of what the software must do after modification is completed. It documents software functionality but does not detail the implementation of the functionality.

Each time the software is updated resulting in a version that is shared internally or tested, either adding a capability or fixing bugs, a Requirements document describing the code modifications to be performed must be completed.

6.3 Implementation Document

Implementation details are documented on the Software Implementation QA form. This includes a comprehensive list of all enhancements and bugs addressed in a new software version and a brief description of the implementation. Any changes that are visible to the end user must be described. This document includes a suggested test plan and acceptance criteria.

The development programmer has the flexibility to design the implementation in the manner he/she believes best.

A Software Implementation QA form is associated with each version of the software that is placed in the revision control system for testing. Incremental versions that may only be partially complete and are not to be used for testing do not need to have an Implementation form.

6.4 Test Plans

Test plans describe how the software is to be tested to show conformance with the software requirements. All MACCS software needs be tested according to a test plan. The test plan may include a set of regression tests if requested by the project manager. Regression testing is generally expected when the version will be distributed to the public or used for a significant application that is to be documented in an externally distributed report. All test plans are documented in the QA documents. These are the Software Regression Checkout QA form and a version specific Software Testing QA form.

6.5 Source Code

All source code must be stored and tracked in the Configuration Management System. Software developers are responsible for checking in modified code and keeping the system current. A secondary, backup copy of source code is to be maintained on a separate server.

6.6 Sample Files

The standard set of WinMACCS sample project files must be stored with the distribution software.

6.7 Data Files

Food model files, created by COMIDA2, and dose conversion files are installed with WinMACCS and can be found in the installation folder. The decay chain file, Indexr.dat, is used by MACCS and is also placed in the installation folder. The census and county economic data files are installed with SecPop and can be found in the installation folder. Common site files containing site latitudes and longitudes are also installed with SecPop and can be found in the installation folder.

6.8 User's and Models Manual

A Users' and Models Manual should be created for new software and maintained for existing software. Information needed for a user to use a code may be in the form of a report, readme file, letter to the customer, etc., or a combination of these. The manual should contain detailed instructions on how to install, configure, and use the software. The manual should describe the hardware necessary for running the software.

As new models are added, the documentation regarding understanding and using the new models should be stored in the Configuration Management System. This information must eventually be incorporated into the user's and models manual. Thus, separate documents describing new models should only be used on a temporary basis until a more formal Users' Manual is revised.

All reports and manuals must be stored in the Configuration Management System.

6.9 Test Reports

Testing must be documented in test reports. These reports describe the results of tests. Test reports document how the MACCS software suite meets, or does not meet, the test criteria and the requirements.

6.10 Bug Tracking Database

An open source web based bug tracking database based on Bugzilla (http://www.bugzilla.org/) is maintained on a server by Sandia National Laboratories. This implementation, called Melzilla (https://melzilla.sandia.gov/), documents known bugs and suggested improvements. The description, status, priority, SVN revision number, and program version in which the fix appears, as well as other information such as the originator, the person assigned, comments, date resolved, who reported the bug, related files, and more.

6.11 Client Data

Client data are maintained on the server and include a list of clients and contact information.

6.12 Distribution Package

This is a package of files used to create a software installation. This usually contains a setup.exe file that copies and installs the software to the user's computer. This is available as a download for the general user or on a server for Sandia users.

7 Reviews and Meetings

Requirements specification, implementation and test plan reviews are designed into the quality assurance documentation. Design and code reviews may be informal and held on an as-needed basis. They may be attended by the whole team, or only by affected members of the team. Documentation of these reviews should be kept in the Configuration Management System.

A secretary should be assigned to team planning meetings. Minutes of the meeting should be place into the Configuration Management System. Some meetings are informal between individual team members. These do not need to be documented.

The quality assurance documents should be reviewed each year.

8 Testing

Verification testing ensures that the program correctly solves equations as intended. Appropriate verification tests are developed and documented. This testing includes module-level (unit) tests, integral tests, and regression tests. Problem reports must be maintained, documenting the results of these tests as decided by the PI or developer.

8.1 Unit Tests

Program developers perform unit tests. These tests verify the performance of new models or functionality. Program developers are responsible for writing any code needed to test their modules. Such code is considered part of the implementation. Module test code should be included with source code, and designs should incorporate module level testing if applicable.

8.2 Internal Release Tests

Release testing must be performed to test software developed by the MACCS2 programmers. As new versions of various components become available, team members should integrate these components into their suite of project software. The MACCS2 team members should perform informal integration testing after modules have been put into the Configuration Management System and are released for internal testing. Not all versions of the software in the Configuration Management System must be tested. Team members should communicate requirements for testing during and following development. The purpose of this testing is to find bugs and improve the user interface before release testing of the software suite.

As bugs are found, they should be entered into the Bug Tracking Database (Section 6.9 or Melzilla).

Formal integration testing of internal releases is not planned, but is done as a team development effort.

8.3 Alpha Testing Phase

Alpha-level testing should be done by people chosen by the PI in preparation for Beta testing. This testing is informal but any defects noted during this testing should be entered into the Bug Tracking Database (Melzilla).

8.4 Beta Testing Phase

As appropriate, testing should be done outside of the MACCS2 development team, as chosen by the NRC, in preparation for a public release. This testing should be documented as defects recorded by each Beta tester. Testing results and comments should be entered into the Bug Tracking Database. A beta testing phase has been much less common than an alpha testing phase during past development.

8.5 Reports and Corrective Action

8.5.1 Reporting a bug

Testers finding problems or bugs during testing should communicate problems or suggestions to the MACCS2 team or the PI, or enter the bug into Melzilla.

All information relating to a particular bug (or a request for improvement) should be input into the Bug Tracking Database. At each stage of reporting, there should be sufficient information to allow another individual to follow up as well as allowing the project manager to judge the adequacy of the investigative effort.

A log number is assigned to the bug by Melzilla. This number is used to reference bugs in documentation and software comments.

8.5.2 Problem Review

The PI reviews new problems. The problem may be referred to a developer for investigation. The problem may be discussed at the next project meeting. Alternatively, the problem report may be closed. Regardless of the action taken, the problem and its status should be documented in the Bug Tracking Database.

8.5.3 Investigation of Problems

Program developers may be asked to investigate a problem. If the problem is simple, the program developer may fix the software and document the results. For complex problems, the solution should be identified and time estimates made. Solutions and estimates are documented in the Bug Tracking Database, and are passed back to the PI for approval, which may involve discussions between the PI and the NRC PM.

8.5.4 Problem Closure

Final resolution of problem reports should be documented, and the issue marked as resolved. Problems can be fixed or not. The reasons a problem is not fixed should be documented in the Bug Tracking Database. Any resolution that has resulting in source code modifications is documented in the QA Design and Implementation forms.

9 Tools

9.1 Tools

The following table describes the tools used on the project:

Tool	Document	Comments
Word 2007 or 2010	All non-source code documents	
ASCII editor	Source code files	Any ASCII editor may be used. Notepad and TextPad are currently being used.
Visual Studio 2008, Intel Fortran 11.1.051	Fortran files	

Microsoft Visual Basic 6 SP5	VB project files	
Microsoft Access 2007 or 2010	Database files	
Microsoft Visual SourceSafe 6.0c	Configuration management tool	Source code and documents must be stored in the SourceSafe repository through October 2012.
Subversion (SVN)	Configuration management tool	Code and documents must be stored in the SVN repository starting in October 2012.
Wise Installation System 9.0	Install package builder.	
Windows XP or Windows 7	All	Operating System
Adobe RoboHelp 7	Help files development tool.	Replaced Word 2000 RoboHelp
Adobe Acrobat	Used to create pdf files	

10 Configuration Management

Subversion (SVN) is the current tool used to control configuration management. All QA documents (including source files) developed for the MACCS2 project are under configuration control. All previous versions of software (prior to the implementation of SVN) are to be maintained on a server computer outside of the configuration management software.

10.1 Server

The current server for the MACCS project is a computer on the Sandia Internal Network named \\snl\ne_gs\. QA files are stored in the following directories:
\\snl\ne_gs\ArcticXP\Maccs Common\ and \\snl\ne_gs\ArcticXP\Maccs Databases\

Document versions are controlled using a “check-in policy”. Team members work on software locally on their own computers. Therefore, documents are not be controlled when they are checked out. A document is controlled when it is checked in to the common area.

The check-in policy is implemented using Apache Tortoise SVN, an interface to the SVN database in the following location:
https://meldevlnx32.sandia.gov/svn/developers/Maccs Common\

Each team member is required to “check-in” his/her software whenever a major modification or a series of minor modifications are completed. Comments should be included at the time of check-in to document changes. The purpose of the Configuration Management System is to provide traceability and reproducibility. It provides a necessary means for retrieving a version previous to current changes.

10.2 Version Numbering

Version numbering for the MACCS2 and all other FORTRAN components must use the following format:

Version 1.15.00.02
where:

1	major revision number, incremented by PI approval
15	incremented when released to the NRC or to the public
00	incremented if a special (branch) version was released a particular person or group
02	minor modification number, incremented by the developer and used to track incremental development

A system of version numbers for the WinMACCS and all other Visual Basic components use the following format:

Version 3.1.15
Where:

3	major revision number, incremented by PI approval
1	incremented when released to the NRC or to the public
15	minor modification number, incremented by the developer and used to track incremental development

Appendix A: Requirements QA Form

Work on the implementation should ordinarily begin after the project manager signs this form. This form may be updated and resigned as the work ordered may change during the implementation of the change orders. This document must be finalized before the Implementation QA Form is finalized.

Document Origination Date	
Software Product Name	
Version Modified	
Version Created	
Person Assigned	

Description of Changes Ordered:

	Signatures	**Date**
Programmer		
Project Manager		

Appendix B: Software Implementation QA Form

Directions:

This form documents changes listed on the Requirements QA form.

After the changes have been implemented, the programmer completes the sections titled "Description of Release" and "Test Plan and Acceptance Criteria". If testing is not needed for this version, it should be stated as such in the Test Plan and Acceptance Criteria section. The tester can add additional testing without modifying this document.

The project manager then signs this document, reviewing this document in the process.

The QA administrator verifies that the source code and supporting documents are properly archived.

Document Origination Date	
Software Product Name	
Version Modified	
Version Created	
Date of Version Created	
SVN Revision Number	

Description of Version Created:

Test Plan and Acceptance Criteria:

I have completed the required internal documentation within the source code.

I committed the software changes to SVN in the following directory:

I have updated the "SVN Revision Number" and "Program version in which this fix appears" fields in the bug database at https://melzilla.sandia.gov/ to reflect the SVN number and the version of the software for each relevant bug fixed.

I have verified that the executable can be built from the primary source files from SVN by checking out the proper revision into a separate directory and building the executable file.

I have made a copy of the supporting documents and placed them in the following directory:

I have created a copy of the distribution files and placed it in the following directory:

I have created a copy of the source and placed it in the following directory:

I have created an additional backup in the following location:

	Signatures	**Date**
Programmer		
Project Manager		

I have verified that the supporting documents have been archived as documented above.

	Signatures	**Date**
QA Administrator		

Appendix C: Software Testing QA Form

This form verifies that testing of bug fixes and/or model improvements has been properly performed for a new code version.

The person performing the testing completes the next section after verifying that the supporting documents are clearly written, testing is complete, and test results are properly archived.

The reviewer verifies that the testing is complete, and possibly offers suggestions for additional testing before finalizing.

The project manager then provides final approval for this document.

The QA administrator verifies that all QA documents are properly archived.

Document Origination Date	
Software Product Name	
Version Tested	
Testing Date(s)	
Operating System Used	

Additional Tests Recommended to be added to Regression Test Suite:

Description of Testing done:

Did tests meet the Acceptance Criteria described in the Software Implementation QA Form?

Summary of testing result (include previous version used to compare results, if applicable):

I have documented the test instructions in sufficient detail such that they can be repeated by an independent reviewer.

I have made a copy of the supporting documents including all test input and output files have placed them in the following directory:

I have created an additional backup in the following location:

	Signatures	**Date**
Tester		

I have verified that the test instructions are described in sufficient detail such that they can be repeated by an independent reviewer.

I have reviewed this document and the supporting documents. My findings are as follows:

	Signatures	**Date**
Reviewer		
Project Manager		

I have verified that the supporting documents have been archived as documented above.

	Signatures	**Date**
QA Administrator		

Appendix D Software Regression Checkout QA Form

Before the software is regression tested and goes through final checkout, all improvements to the code must be tested and the Software Testing QA Form must be completed for the new version if required, as indicated in the Software Implementation QA Form.

The reviewer verifies that this document is complete, and possibly offers suggestions before finalizing.

The QA Administrator fills in the next section to verify that the test procedures and supporting documents are clearly written and properly archived.

The project manager then signs this document verifying that the work has been completed.

Document Origination Date	
Software Product Name	
Version Tested Against	
Version Tested	
Testing Date(s)	
Operating System Used	

Additional Tests and Modifications to Regression Suite:

Auxiliary Software Used for Testing:

Related Documents:

Description of Testing done:

Summary of testing results:

I have documented the test instructions in sufficient detail allowing recreation by an independent reviewer.

I have made a copy of the supporting documents including all test input and output files have placed them in the following directory:
I have created an additional backup in the following location:

	Signatures	**Date**
Tester		

I have verified that the test instructions are described in sufficient detail such that they can be repeated by an independent reviewer.

I have reviewed this document and the supporting documents. My findings are as follows:

	Signatures	**Date**
Reviewer		
Project Manager		

I have verified that the supporting documents have been placed in the directory as documented above.

	Signatures	**Date**
QA Administrator		

Appendix E: QA Training Completion Form

A team member completes the quality assurance (QA) training by reading and completing instructions in the document "QA Training.docx". When this process is completed, the completion date of the training, the version and SVN number associated with the training document, the member's signature, and finally the QA administrator's signature needs to be completed.

Each team member is responsible for completing the training.

Periodically the training needs to be repeated. When this is done, a new entry is made in this document.

Team Member Name:

	Version:	**Signatures:**	**Date:**
Team Member:			
QA Administrator:			

	Version:	**Signatures:**	**Date:**
Team Member:			
QA Administrator:			

	Version:	**Signatures:**	**Date:**
Team Member:			
QA Administrator:			

Appendix F: Data Verification QA Form

This form verifies that the contents of the data set of interest have been verified to be consistent with the data requirements. Requirements can be defined in this document.

The person performing the testing completes the next section after verifying that the supporting documents are clearly written, testing is complete, and test results are properly archived.

The reviewer verifies that the testing is complete, and possibly offers suggestions for additional testing before finalizing.

The project manager then provides final approval for this document.

The QA administrator verifies that all QA documents are properly archived.

Document Origination Date	
Software Product Name that Uses Data	
Version Tested	
Testing Date(s)	

Description of Testing done:

Summary of testing result (include previous version used to compare results, if applicable):

I have documented the test instructions in sufficient detail such that they can be repeated by an independent reviewer.

I have made a copy of the supporting documents including all test input and output files have placed them in the following directory:

I have created an additional backup in the following location:

	Signatures	**Date**
Tester		

I have verified that the test instructions are described in sufficient detail such that they can be repeated by an independent reviewer.

I have reviewed this document and the supporting documents. My findings are as follows:

	Signatures	**Date**
Reviewer		
Project Manager		

I have verified that the supporting documents have been archived as documented above.

	Signatures	**Date**
QA Administrator		