Quality Assurance Project Plan

SWMM Redevelopment

by

Trent Schade, Project QA Manager

November 27, 2002

This document has been reviewed and endorsed in accordance with the U.S. Environmental Protection Agency's Quality Assurance Project Plan review policies.

Signatures

Signed	Nov 26, 2002
Trent Schade, Project QA Manager	Date
Signed	Nov 26, 2002
Tom Wagner, NRMRL QA Officer	Date

National Risk Management Research Laboratory
U.S. Environmental Protection Agency
26 W. Martin Luther King Drive
Cincinnati, OH 45268

Contents

Contents i
List of Figures ii
List of Tables ii
Acronyms and Abbreviations iii
Distribution List iv

1.0 Project Description 1
1.1 Project Description and Schedule 2
1.2 Quality Objectives 2
1.3 Project Organization 3
2.0 Functional Requirements 3
2.1 Sampling Method Requirements 3
2.2 Acceptance Criteria 6
2.3 Analytical Method Requirements 6
2.3.1 Testing Plan 7
2.3.2 Testing Implementation 9
3.0 Quality Control Requirement 11
3.1 Testing Requirements 11
3.2 Software and Hardware Requirements 12
3.2.1 Development Environment Requirements 12
3.2.2 Test System Requirements 12
3.2.3 Version Management System Requirements 13
4.0 Data Acquisition Requirements 14
4.1 Data Management 14
4.2 Documentation and Records 14
4.2.1 Code Standard 15
4.2.2 Defect Report Requirements 17
5.0 Assessments and Response Actions 17
5.1 Data Review, Validation and Verification Requirements 17
5.2 Validation and Verification Methods 17
5.3 Reconciliation with Data Quality Objectives 18
6.0 Maintenance and User Support Requirements 18
7.0 Reports to Management 19
References 20

Appendices
A. Mapping of WSWRD QA Requirements to Software and Data Management QAPP Elements
B. Standard Operating Procedures

Attachments
A. SWMM Project Development Plan

List of Figures

Figure 1. Testing Procedure 9

Figure 2. Example Header File 15

List of Tables

Table 1. Staff Descriptions 4

Table 2. Computational Features Contained in SWMM 5

Table 3. Non-critical Metrics 6

Table 4. Product Formats 14

Table 5. Defect Report Fields 16

Acronyms and Abbreviations

BMP	Best Management Practice
CDM	Camp Dresser and McKee Inc.
CPU	Central Processing Unit
CHI	Computational Hydraulics International
CRADA	Cooperative Research and Development Agreement
CVS	Concurrent Versioning System
DLL	Dynamic Link Library
DWF	Dry Weather Flow
EMC	Event Mean Concentration
EPA	U.S. Environmental Protection Agency
GUI	Graphical User Interface
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
LID	Low Impact Development
OOP	Object Oriented Programming
RAM	Random Access Memory
SOP	Standard Operating Procedure
SWMM	Storm Water Management Model
QA	Quality Assurance
QAPP	Quality Assurance Project Plan
WSWRD	Water Supply Water Resources Division
UWMB	Urban Watershed Management Branch
WQMB	Water Quality Management Branch

Distribution List

Name and Title	Organization
Sally Gutierrez, Division Director	EPA, WSWRD, Cincinnati, OH
Tom Wagner, NRMRL QA Officer	EPA, NRMRL, Cincinnati, OH
Dan Sullivan, Branch Chief, SWMM Redevelopment Project Officer	EPA, WSWRD, UWMB, Edison, NJ
James Goodrich, PhD., Branch Chief	EPA, WSWRD, WQMB, Cincinnati, OH
Dennis Lai, PhD., Environmental Engineer	EPA, WSWRD, UWMB, Edison, NJ
Lewis Rossman, Ph.D., Environmental Engineer	EPA, WSWRD, Cincinnati, OH
Edward Burgess, Project Manager	CDM, Cincinnati, OH
Robert Dickinson, Ph.D., Principal Investigator	CDM, Tampa Bay, FL
Carl Chen	CDM, Cincinnati, OH

1.0 Project Description

The EPA Storm Water Management Model(SWMM) is a dynamic rainfall-runoff simulation model, used for single-event or long-term (continuous) simulation of runoff quantity and quality from primarily urban areas. Applications of SWMM in the engineering community are numerous, but the current implementation of SWMM has a very steep learning curve that limits its appeal to younger engineers and engineering educators. The current version of SWMM includes thousands of lines of FORTRAN code much of it written many years ago and updated sporadically. This project attempts to gain some quality control over the software development process while at the same time redeveloping and modernizing the basic functions of the software. As development of the software continues in the future, these quality standards will be maintained along with the source code and documentation. Attachment A includes the SWMM Redevelopment Plan which describes the following objectives of this project:

- To revise the architecture of the SWMM computational engine, using object oriented programming (OOP) techniques, to enhance the ability of the model to be maintained, upgraded, and interfaced with other software.

- To provide a rudimentary graphical user interface (GUI) to the engine to improve the usability of the model.

- To remove obsolete features, improve key computational aspects, and add new computational capabilities to the model where warranted.

- To develop guidelines on how SWMM can be used to model more recently developed Best Management Practices (BMP) for runoff control.

The project will produce the following:

- A newly coded version of the SWMM (SWMM 5.0) computational engine that can be run either as a stand-alone application or as a Dynamic Link Library (DLL) of functions that can be called from other applications such as third party vendors of SWMM.

- A GUI shell program that will run under Windows, access the SWMM engine through DLL calls, and include a context-sensitive, on-line Help system.

- Full documentation in the form of a Users Manual, a Reference Manual, and a Programmer's Manual,.

- A manual on Modeling BMPs with SWMM that will illustrate how SWMM can be used to model various types of BMP/LID options.

This Quality Assurance Project Plan (QAPP) describes methods and procedures to test the quality of the redeveloped code. It also describes requirements for documentation, version tracking, and long-term maintenance. The Modeling BMPs project is under a different contract for which there is a separate QAPP.

1.1 Project Description and Schedule

This QAPP addresses the problem of maintaining a record of the quality practices applied during the current SWMM redevelopment project and future related software development. The QA portion of this plan includes software testing, documentation standards, version control procedures, and long-term maintenance.

This QAPP does not attempt to address the requirements for the application of SWMM to a particular engineering problem. The QA for applications of the revised SWMM is best addressed on a case-by-case basis. EPA's Council on Regulatory Environmental Modeling provides the latest information on model development and application on their website, **www.epa.gov/ord/CREM**. Projects to collect data for input to a SWMM model should also provide a QAPP for the data collection effort.

This project is on a very rapid schedule. Much of the initial design work is complete, and the entire project will be complete by March 2003. Additional description and schedule for this project are in the SWMM Redevelopment Plan included as Attachment A.

1.2 Quality Objectives

EPA has guidance for planning project quality objectives in three areas concerning environmental modeling: software implementation, model application, and model development (Reilly, 1993; Browner, 1994; USEPA, 2000a and b; USEPA 1995).

The term "software implementation" refers to projects involving software development, software/hardware systems development, database design and maintenance, and data validation and verification systems. The "software implementation" guidance is the basis for this QAPP. It is rearranged according to guidance provided by the Division QA Manager to meet WSWRD requirements. The table included as Appendix A shows how the software implementation guidance maps to the division's QA requirements.

"Model application" refers to projects that utilize a well-established and well-verified environmental model to solve a specific environmental problem. SWMM applications, especially those that inform regulatory decisions, should include a QAPP that follows the EPA guidance on environmental model application.

"Model development" describes a project to verify and validate a numerical representation of a physical system. This project begins with the premise that the existing implementation of SWMM is well-developed. The model development of SWMM is published in peer-reviewed journal articles over the 30-years since SWMM's inception. Future model development should be done under the EPA guidance for basic science that includes rigorous peer-review for all publications.

The overall quality objective for this project is to re-program SWMM into modernized code with a demonstrated level of quality in the source code and documentation. The new version of SWMM will incorporate the best practices of the software development community.

1.3 Project Organization

This project is a joint development effort between EPA-NRMRL's Water Supply and Water Resources Division (EPA) and Camp Dresser and McKee Inc. (CDM). CDM's participation is through a Cooperative Research and Development Agreement (CRADA) between itself and EPA. The project also solicits input from others in the SWMM community.

Two experienced software developers, a principal investigator from EPA and a principal investigator from CDM, will produce most of the redeveloped code. Responsibilities of the EPA principal investigator include redesigning SWMM to be object oriented, implementing the design for the computational engine using C++, and adopting an existing graphical user interface (developed previously for pressurized pipe networks) for use with SWMM. The responsibilities of the principal investigator for CDM will include redeveloping the EXTRAN module of SWMM, developing input data conversion programs for previous versions of SWMM, and standard operating procedures (SOP) development. Table 1 lists the roles and responsibilities of EPA and CDM staff.

EPA and CDM have created a Technical Review Panel to provide guidance and review of the SWMM redevelopment project. The panel consists of noted SWMM experts representing different segments of the SWMM user community. The panel will serve two functions. One is to suggest what the form and function of the redeveloped SWMM should be. The second is to review the software during the Beta test phase of the testing plan.

2.0 Functional Requirements

This section describes the functional requirements for this project. EPA and CDM have already held a meeting of the Technical Panel to discuss this subject. Table 2 summarizes the results of this meeting. The panel suggested that only very modest changes be made to the current functionality of SWMM.

2.1 Sampling Method Requirements

This section does not apply to this project. Any application of the SWMM model that includes water quality monitoring or testing should include a QAPP specific to that application. The sampling plan should consider temporal, spatial, sample aggregation, and sampling methods.

Table 1. Staff Description

EPA Staff	Role	Responsibility
Daniel Sullivan	Project Manager	Project Schedule Manage Contracts
Lewis Rossman	Principal Investigator	Requirements Specification C++ Code Development Design Review/Changes Unit Testing
Dennis Lai	Project Engineer	Technical Review
Trent Schade	Project QA Manager	QAPP development Testing after Unit Testing Version Tracking
	Division QA Manager	QA Review
CDM Staff	**Role**	**Responsibility**
Edward Burgess	Project Manager	Manage CDM Involvement Technical Review
Robert Dickinson	Principal Investigator	EXTRAN FORTRAN Code Development Technical Review Testing Design Review/Changes
Charles Rowney	Project Manager	Technical Review
Technical Review Panel	**Technical Guidance Project Review**	**Describe SWMM Functional Requirements Review SWMM during system testing**

Table 2. Computational Features Contained in SWMM

Rainfall		**Collection Network Flow**	
User-Supplied Time Series	U	Nonlinear Reservoir	**X**
Current NWS File Formats	U	**Transport Network Flow**	
Older Formats	U	Kinematic Wave	U
Event Statistics (SYNOP)	U	Explicit Dynamic Wave	U
Infiltration		*Implicit Dynamic Wave*	■
Horton	U	**Pollutant Loading**	
Green Ampt	U	Buildup Functions	U
SCS Curve Number	■	Washoff Functions	U
Rain Dependent Infiltration	U	Street Cleaning	U
Transport Infiltration Input	**X**	Catchbasins	**X**
Groundwater		Dust and Dirt	**X**
Two-Zone Model	U	Erosion	U
HEC-HMS SMA Model	■	DWF Estimation	**X**
Snowmelt		**Pollutant Routing**	
Degree Day Model	U	CSTR Model	U
Energy Balance Model	●	*Lagrangian Transport*	■
Evaporation		**Sediment Transport**	
Monthly / Daily Time Series	U	Shields Scour/Deposition	U
Meteorological Data		*Other Scour/Deposition*	■
Daily Time Series/NWS File	U	*Bed/Suspended Load*	■
Overland Flow		**Storage/Treatment**	
Nonlinear Reservoir Model	U	End-of-Pipe	U
Special Drainage System		*Inline*	U
Pumps, Orifices, and Weirs	U	**Real Time Control**	
Flow Dividers	U	Level Control	U
Storage Units	U	Timer Control	U
Backwater Elements	**X**	*Rule-based Control*	●
Natural Channels & Bridges	U	*Interactive Control*	●

U feature exists in current version of SWMM
+ feature should be added to SWMM
? undecided if feature should remain or be added to current version of SWMM
X delete feature from SWMM
● feature should be added to SWMM, however, it is a low priority
■ feature would improve SWMM, but its incorporation is on hold

2.2 Acceptance Criteria

A battery of test cases will be developed that exercises the functions shown in Table 2. Each test case will include information about the part of the software that it tests.

Identical test cases will be developed for the new version of SWMM, SWMM 5.0, and the existing version of SWMM, SWMM 4.4. Our goal is to meet the following numerical criterion for each point of the outflow and depth hydrograph at all points of the drainage network in each test:

$$\frac{|A_t - B_t|}{B_t} \leq 0.005, \quad for\ |B_t| \geq 0.005$$

$$|A_t| \leq 0.005, \quad for\ |B_t| < 0.005$$

Where A_t represents the depth and flow from the SWMM 5.0 simulation at time t, and B_t represents the depth and flow from SWMM 4.4 simulation at time t. Roughly speaking, this requires that all numerical results from the two versions of SWMM agree to three significant figures. Prior to release, any unresolved excursion from this criteria will require a full explanation and accounting in the SWMM 5.0 documentation.

Another measure of the performance across all test cases is the system continuity error. The system continuity error is a measure of the volume of water gained or lost by numerical problems. Small discontinuity, less than 2.0%, is expected. Large discontinuity indicates either a problem with solution time-step or a problem with the program. Each test case will compare system continuity error with system continuity error from the previous version of SWMM.

Also, each test case will describe a series of non-critical statistics that will help the Review Panel determine performance of new SWMM versus existing SWMM. It may be desirable to develop metrics for system performance, function performance, or code quality. The following table shows examples of these metrics.

Table 3. Non-critical Metrics

System Metrics	Function Metrics	Code Quality
• CPU time. • Actual Run time. • Number of iterations. • Node continuity.	• Flow statistics • Velocity statistics • Hydrograph error • Pollutograph error • Mean event concentrations • Time to peak • Baseflow	• Number of comment lines divided by total number of lines.

2.3 Analytical Method Requirements

Analytical methods for this software implementation include requirements of the testing procedures. These requirements are described below as the testing plan and the testing plan implementation.

2.3.1 Testing Plan

The testing plan includes the following six levels of testing.

- Unit/Module Testing
- Build Test/Smoke Test
- Regression Testing
- Acceptance Testing
- System Testing
- Beta Testing

The following paragraphs define each level of testing including the individuals responsible for the testing, the frequency of the testing, the reporting requirements for the testing, and the required response to defects.

Unit/Module Testing

Individual developers who are responsible for writing a unit or module may perform Unit/Module Testing at their own discretion. When a developer considers unit or module testing necessary, the developer will make the test program a utility for the build. This enables future development or redevelopment of the module or unit to utilize the same testing program. The test program should follow all applicable standards for project code and source code control; however, it will not be documented in the reference manual or users' manual. EPA is considering using an open source testing framework such as CppUnit (http://www.xprogramming.com/software.htm).

Build Test/Smoke Test

All developers shall submit code to the QA manager for build testing and smoke testing on a weekly schedule. They will use a makefile approach to the component build. The build test is simply to test that the revisions to any modules allow the source code to compile. This test should also include testing of any installer software. The smoke test runs several simple functions several times to make sure there are not basic problems (e.g., memory leakage) with the basic functionality. The QA manager will report the results of each build test/smoke test to the developers as either a pass or a report of errors/exceptions from the compiler or the operating system. EPA is considering using a software tool such as Electric Fence or Purify that will automatically check compiled software for improper use of memory.

Regression Test

To facilitate early testing of the software, the regression testing will be performed on each module as it is completed in the following order:

1. Runoff Hydraulics
2. Runoff Water Quality
3. Kinematic Wave Routing
4. Kinematic Wave Water Quality
5. Dynamic Wave (Extran) Routing
6. Dynamic Wave Water Quality
7. Rainfall File Reader
8. Continuous Event Simulation
9. Rainfall/Runoff/Routing Statistics

As modules are completed, the regression test ensures that they work within the construction of the full package. The regression test will run a battery of several input files through the computational engine exercising individual hydrologic processes as indicated in Table 2. The QA manager will develop an automated script to run the regression tests and parse the output to perform the comparisons. Also, the QA manager will perform the regression tests and report results to the development team. The regression test will be repeated for each module as it is completed or updated.

In communication with CDM and the Technical Panel, EPA QA staff will prepare a battery of test cases as input files that test the system behavior. The input files are designed to test various computational features of the program as described in Table 2. The input files for SWMM 5.0 will build on a suite of files used by CDM for testing previous versions of SWMM. A fact sheet will describe each test case. The fact sheet will include a schematic of the test system, the theory behind the system behavior, the parts of the model under review, critical output and non-critical output. The test cases and the associated fact sheets will be released with the User's Manual.

Acceptance Test

The acceptance test runs the same battery of test cases as the regression tests through the computational engine. Only after the majority of the units or modules are functional will the QA manager perform the acceptance tests. The results of the acceptance tests will show instances where modules produce strange results, and it will show which module passed values that initiated errors in other modules. The QA manager will report results to the development team, and the QA manager will document all results in the version control system.

Data generated from the QA testing should present the same information for both SWMM version 4.4 and SWMM version 5.0. Different applications of SWMM have different needs with respect to most of these metrics, so it is up to the individual application developers to determine acceptance testing standards.

Data collected for QA will describe performance for a suite of core functions available in SWMM 5.0. The same functions will be tested in SWMM 4.4. Comparison with SWMM 4.4 provides users with information about relative impact of the upgrade on performance of individual functions.

System Test

To facilitate early detection of inconsistencies or problems with the GUI, the system testing will be ongoing throughout the development. This test will exercise each function of the software available through the GUI. The QA manager will develop this test SOP from the design of the GUI. The QA manager will present any negative results from this test to the development team for review. An acceptable test (no defects found) will move the code from pre-beta into beta. Given that perfect code is not really attainable on large projects, minor defects will be documented in the version control system as a "bug fix" list. Minor defects include aesthetic concerns, formatting, colors, graphics, unexpected or non-intuitive labeling. Major defects include system crashes, long delays, incorrect labeling, or missing functions.

Beta Testing

Once the system test produces relatively error free code, SWMM 5.0 will move into beta-testing. Beta testing will occur in two phases, the Review Panel phase and the community group phase. At this point the software will be presented to the members of the Review Panel. These individuals will test the software and report any problems or errors that they encounter.

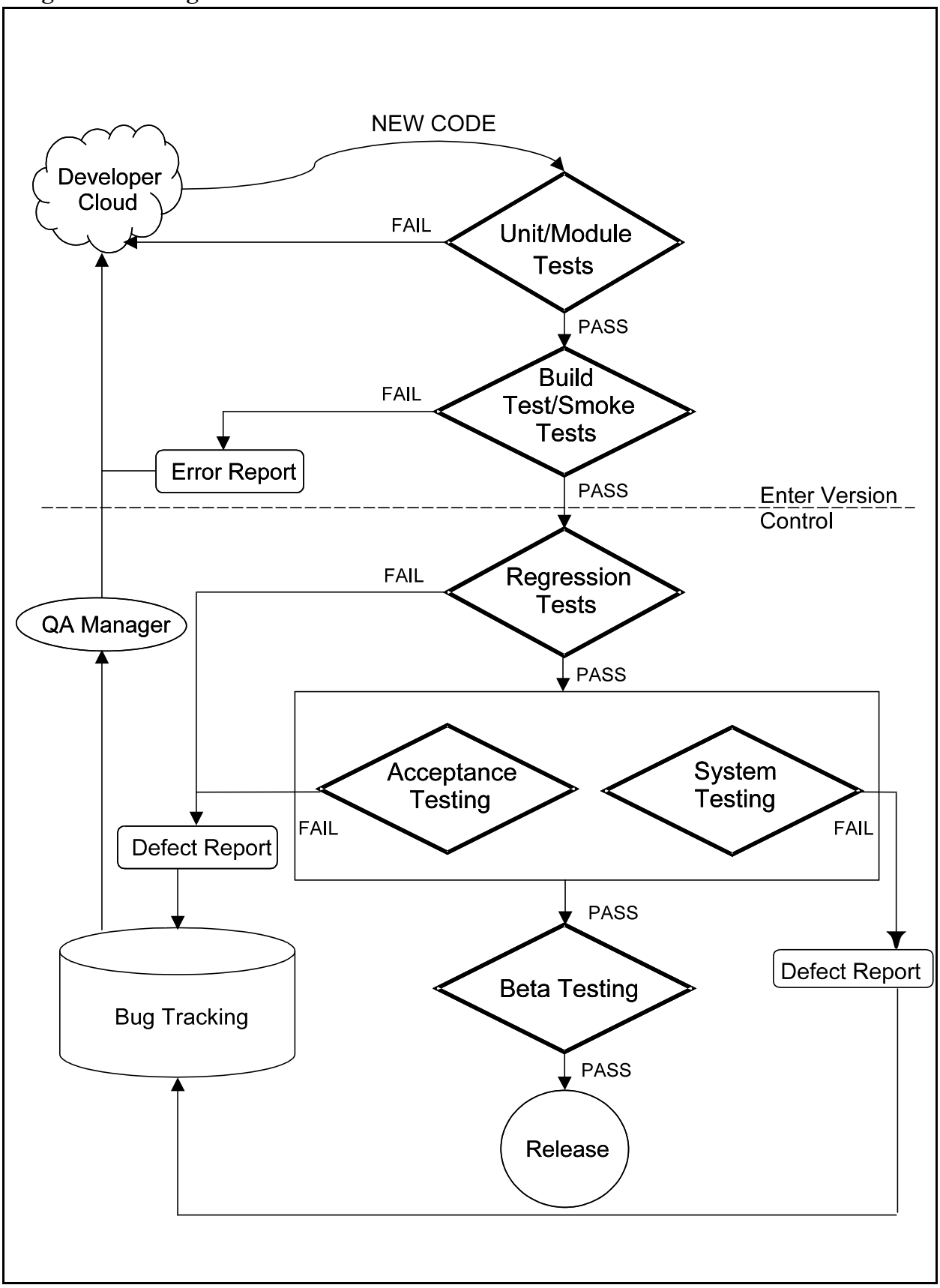
The second phase of Beta testing will present the software to the SWMM community for testing. This project will utilize an existing community list server with about 400 subscribers created and moderated by Professor William James of Guelph University. The list server is available at http://www.chi.on.ca/swmmusers.html. The community has a long history of intense involvement in the development and testing for previous versions of SWMM. When SWMM 5.0 goes into beta testing, EPA will consider making the version control system become publicly accessible through a free web-based change request tracking tool. Also, the bug tracking database will be made publicly available. This will make all bug tracking and release testing a community process.

2.3.2 Testing Implementation

Figure 1 shows a flowchart for the entire testing procedure. Detailed procedures for each of the tests are described in Standard Operating Procedures (SOPs). The SOPs provide sufficient detail for SWMM users to duplicate these tests.

The testing will be implemented according to the SOPs described in Appendix B. Details of these SOPs are subject to revision which will be maintained along with the QAPP.

Figure 1. Testing Procedure



3.0 Quality Control Requirements

Quality control requirements describe the implementation of the quality systems. The following sections describe requirements for the testing systems and the hardware and software systems needed to implement the quality control.

3.1 Testing Requirements

This section describes the requirements under which the software passes each of the above tests. In general, the tests are pass or fail.

Unit/Module Tests: This test is totally at the Programmer's discretion. The suggested criteria are that all of the features and functions of the module interface properly with the unit test program. Also, the code should implement all of the features and functions required by the design. The programmer decides when the module is ready to add to the build test.

Build Test/Smoke Tests: The code must compile with no errors on the QA Manager's system utilizing a makefile build process. The executable program must be able to receive and process user input and produce a result in a human readable format. Results from this test are lists of errors sent to the developer, or the code goes to version control.

Regression Tests: Modules modified on the developer's system must compile and run under the version controlled system. The battery of test cases must produce results consistent with previous versions under version control. Failed tests should result in a defect report which will be sent to the developer and tracked by the QA manager. The module cannot undergo acceptance testing until defects found here are corrected. Developers should strive to correct defects in critical code within one week.

Acceptance Tests: Results from the battery of test cases should match according to a panel of experts results from previous versions of SWMM, or the results should match a data set with a known level of quality. Failed acceptance tests should result in a defect report which will be sent to the developer and tracked by the QA manager. Defects found here should be corrected within one week. However, the potential exists for defects at this level to be the result of analysis techniques or design decisions that may take longer to remedy.

System Tests: System testing may be conducted parallel to the Acceptance Tests. All of the features available on the GUI should work. A failed test results in a "bug". The bug will be reported to the developers and tracked by the QA manager.

Beta Tests: An initial round of Beta testing will be limited in time and number of testers. The final round of Beta testing will distribute the code to a wider audience, and basically remain open until the final version is released. All bugs uncovered during beta testing will be tracked by the QA manager. Bugs at this point will be prioritized by the EPA principal investigator and remedied as soon as possible.

The QA manager will keep statistics on defect reports and bug reports. The statistics will show which modules are most likely to undergo repair or modification. Eventually, the statistics should

show which modules may be ready to be redeveloped. Also, statistics will help plan future development efforts.

3.2 Software and Hardware Requirements

This software redevelopment effort includes requirements for three systems: development environment, test system and version management system

3.2.1 Development Environment Requirements

For the solution engine, developers at the U.S. EPA will use standard C++ with no special libraries that require commercial software licenses to compile or run. For rapid development and debugging EPA researchers might utilize an integrated development environment (IDE) such as Borland's C++ Builder. The build procedure from within this IDE is somewhat modified from publicly available compilers and debuggers. EPA will provide a basic "make" file that works on a public compiler like gcc and a public debugger like gdb. The compiler will specify options to use only the ANSI standard language.

The GUI will be developed with tools available from Borland's Delphi compiler, and EPA will not provide a public compilation support or tools for the GUI. Basically, the GUI must be installed using an install program provided with the software. However full source code for the GUI will be provided.

3.2.2 Test System Requirements

All tests described above have similar requirements for the test system. All computers used for testing will have at least the following specifications:

A 32-bit CPU clocked greater than 200 MHz.
512 Kbyte on-board L2 cache.
64 Mbyte RAM.

During development the computers used for testing shall be consistent; most likely the QA manager will use the same computer for all tests. The operating system employed by the testing machine is Microsoft Windows 98 SE. The SOPs for each test (Appendix B) spell out procedures to ensure consistency in operations of the systems and recording system parameters.

For the build and smoke tests, the computer for testing should have previous versions of the software removed. This ensures that the version under test is the correct version, and it ensures that the test version installs completely.

There is no real performance benchmark needed for the computer on which the tests will be run. After the beta release, the project will rely on the community-at-large to provide feedback on performance within alternative operating systems.

3.2.3 Version Management System Requirements

Software and test file version management will be performed using a freely-available file managment system called Concurrent Versions System (CVS). This software is hosted on a Linux-based computer connected to the QA Manager's system through a LAN. The CVS software allows updates from multiple users to the repository of text based files. This repository will track all changes made to the code and to the input files. Reverting to previous version is very simple using CVS.

According to Figure 1, files may enter version control only after a successful build test. Therefore, prior to the build test, the developers are responsible for maintaining a set of files for their development activities. Adding or modifying a module requires a build test on the developer's code. After a successful build test the developer's files are compared to the current files. All files from the developer's directory that are different from the current files are moved into the test directory. The program is then compiled according to the SOP for the regression test.

The following file structure provides the necessary version control and data management to meet all of the software maintenance and testing requirements. Items in bold will be included in the version management system. Additional directories will be added as needed to the version control system.

SWMMRedevelopment/
- **swmm5/**
  - **src/ {*.cpp; *.h; makefile}**
  - lib/ {*.o; *.so; *.dll}
  - bin/ {*.exe}
  - **docs/**
  - **input/ {*.inp}**
  - output/ {*.out}
- **swmm4/**
  - **src/ {*.for}**
  - lib/ {*.o; *.so; *.dll}
  - bin/ {*.exe}
  - docs/
  - **input/ {*.inp}**
  - output/ {*.out}
- **QAtests/**
  - **smoke/**
    - **perl/ {*.pl}**
    - **control/ {*.txt}**
    - **results/ {*.txt}**
  - **regression/**
    - **perl/ {*.pl}**
    - **control/ {*.txt}**
    - **results/ {*.txt}**
  - **acceptance/**
    - **perl/ {*.pl}**
    - **control/ {*.txt}**
    - **results/ {*.txt}**

4.0 Data Acquisition Requirements

This project relies on previously collected data. CDM and the Technical Panel developed the test cases through years of research and product testing new versions of SWMM. The quality of the past data is based on years of experience with the application of these models. This data was collected for specific evaluations of new processes installed in SWMM.

Input files for the existing versions of SWMM will be replaced. The QA manager will generate the test files in the new format using software developed by CDM. SWMM also relies on data from external sources such as the National Weather Service or the USGS. Testing for the Rainfall Data Reader Module will primarily focus on the accuracy of reading external data files.

EPA may utilize an industry standard file format such as XML for input files or data files. In that case, EPA will utilize code that is certified by the appropriate standard organization for parsing input or data files to ensure compliance with the industry standard.

4.1 Data Management

The QA manager will be responsible for tracking changes to all data, source code, and work products. The QA manager will provide a facility to archive and back-up data related to this redevelopment effort. Weekly backup of all project files will be completed by writing the data to a CD-RW on the QA manager's computer. The data will be moved off-site monthly. The off-site storage facility will maintain monthly back-ups for the duration of the project.

4.2 Documentation and Records

All records relating to the development of this software will be kept in accordance with the OARM requirements. In addition, the records used for this project will be used as raw data for publication in peer-reviewed journals. Documents produced by this project fall into one of four categories: User's Products, Developer's Products, QA Products, and Internal Products. The following paragraphs describe the scope of these products. Table 4 shows the suggested format for these products.

User's products are public documents intended to support the installation and application of the revised SWMM software. These products are the front line documentation for both inexperienced and experienced users, so the quality of these products is critical. The information in the user's manual will also be integrated into the help file system of the software. The user's manual and technical reference will be available on one of EPA's public access websites. They will be under same change control as source code (CVS), and they will be continuously upgraded.

Developer's products will provide information for future improvements or GUI development. These records should follow the best practice for software documentation. They will be under change control. They will continue to change after the release of the latest version of the software is complete.

QA products are internal reports that are required under this QAPP. The quality test runs will return text files as the report format. The standard information will include test case number, name

of tester, test system identification, date, time, elapsed time, critical performance information, non-critical performance information. These documents will be archived with the current version of SWMM under test.

Internal products define the scope of the project, extramural agreements, and quality requirements. These products have limited use outside of EPA. Outputs from this project include a workplan, Technical Panel meeting reports, interim reports, a technical reference, and a user's manual.

Table 4. Product Formats

Products	Electronic Format	Note
User's Products		
Window's Install Files	Windows Executable	
Window's Installation Guide	HTML Document	
Window's Installation Wizard	Windows Executable	
User's Manual	HTML Document	
Application Guide	HTML Document	
Conversion Guide	HTML Document	
Developer's Products		
Source Code	plain text	Standard C++
Programmer's Manual	HTML Document	
QA Products		
Test Cases	plain text	SWMM Input File Format
Test Reports	plain text	Test system output
Defect Reports	plain text	Defect tracking format
Defect Statistics	plain text	
Internal Products		
QAPP	WordPerfect	
Work Plan	WordPerfect	
CRADA	WordPerfect	

4.2.1 Code Standard

All C++ code developed under this plan should follow standard practice of software development. The code should be readable and well commented. At a minimum, each implementation file (example.cpp) should include a header file (example.h). The header file should include definitions, declarations, macros, and inline functions, and the implementation file should include code. Both the implementation code files and the header files should include the following information: Title, Author, Date, Version, History, Purpose, and Usage. Additionally, the files should include References, File Formats, Restrictions, Error Handling, and Notes as necessary. Figure 2 shows an example of comments included in a header file.

Figure 2. Example Header File

```
// ---- runoff.h

//-----------------------------------------------------------------------------
//Required Header
//
// Author: Lew Rossman
// Date: 11/14/2001
// Version: development
// History: built from scratch
//
// Purpose:
// Definition of the RunoffAnalyzer class for EPA-SWMM
//
// Usage:
// This module handles the input and output for the runoff objects.
//
//
//-----------------------------------------------------------------------------
//Optional Header
//
// References:
// Huber, W.C. and R.E. Dickinson. 1988. “Storm Water Management Model
//     Version 4, User’s Manual.” EPA 600/3-88/001a(NTIS PB88-236641/AS).
//     U.S. Environmental Protection Agency, Athens, GA.
//
// Input File Format:
// Not applicable.
//
// Output File Format:
// The writeRunoff will output a text table with the following columns,
// generally this is useful for debugging:
// Time    Land Area        Rainfall  Infiltration  Storage  Runoff
// 10:00   Area 1               0.00        0.0000   0.0000    0.00
// 10:00   Area 2               0.00        0.0000   0.0000    0.00
// 10:00   Area 3               0.00        0.0000   0.0000    0.00
// 10:00   Area 4               0.00        0.0000   0.0000    0.00
// 11:00   Area 1               0.00        0.0000   0.0000    0.00
// 11:00   Area 2               0.00        0.0000   0.0000    0.00
// 11:00   Area 3               0.00        0.0000   0.0000    0.00
// 11:00   Area 4               0.00        0.0000   0.0000    0.00
// Note: writeRunoff writes each area within a time step before moving
// to the next time step.
//
//-----------------------------------------------------------------------------

#ifndef RUNOFF_H
#define RUNOFF_H

//-----------------------------------------------------------------------------
class RunoffAnalyzer    {

private:
    void writeRunoffHeader();
    void writeRunoff(float t);

public:
    void execute();
    RunoffAnalyzer();
};

#endif
```

4.2.2 Defect Report Requirements

Any defects encountered during quality testing performed by the QA manager will be recorded using a standard format. Table 5 shows the standard fields for a defect report.

Table 5. Defect Report Fields

1.	Defect ID	10.	Priority: Low, Medium or High.
2.	Defect Description	11.	Phase created: requirements, architecture, design, construction, testing
3.	Steps to produce defect	12.	Phase detected: requirements, architecture, design, construction, testing
4.	Platform Information: CPU, memory, disk space, video card.	13.	Date Corrected:
5.	Version Information	14.	Corrected by:
6.	Defect's current status: open or closed	15.	Effort (hours) for correction:
7.	Detected by:	16.	Work Product corrected: code, User Manual, Specification Manual, Test Case Files to duplicate error.
8.	Date Detected:		
9.	Severity: cosmetic, serious, or critical		

5.0 Assessments and Response Actions

The Review Panel will review all testing reports, defect reports, and bug-fix lists prior to Beta Testing and prior to public release of the software. The assessment will provide feedback to EPA on the quality of the product and suggestions for improvement. This assessment will not restrict EPA from distributing the software publicly.

5.1 Data Review, Validation and Verification Requirements

Data developed under this project includes instructions to install and interact with the software, guidance to apply the model, details about the quality assurance for the software, and comparison with previous versions of the software. This information will be verified and validated by testing with the user community. The procedure for the beta testing is included in Appendix B.

5.2 Validation and Verification Methods

As each product nears completion, it will be delivered to the QA manager. The QA manager will deliver the product for appropriate agency review as shown in table 5 below.

Table 5. Product Review Requirements

Products	Publication Type[1]	Additional
User's Products		
Window's Install Files	Software	None
Window's Installation Guide	Software	None
Window's Installation Wizard	Software	None
User's Manual	Published Report	Technical Panel
Application Guide	Published Report	Technical Panel
Conversion Guide	Published Report	Technical Panel
Developer's Products		
Source Code	None	None
Programmer's Manual	Published Report	CDM Staff
QA Products		
Test Cases	None	Technical Panel, CDM
Test Reports	None	Technical Panel, CDM
Defect Reports	None	Technical Panel, CDM
Defect Statistics	None	Technical Panel, CDM
Internal Products		
QAPP	Internal Report	None
Work Plan	Internal Report	Technical Panel, CDM
CRADA	Contract Document[2]	CDM

[1]The review requirements for these report types are described in NRMRL guidance documents.
[2]The review requirements for the CRADA are described in ORD Project Officer guidance documents.

5.3 Reconciliation with Data Quality Objectives

The final installation package will be checked by installing on a clean system with updated virus protection, running acceptance tests and system tests, and running virus checkers. The QA manager will compare features available in the final version with features described in table 2.

The final test for the quality of the finished product is acceptance within the user community. This can only be addressed over time. After 6 months of review from the user community, the project team will review the public response to the redeveloped SWMM and develop a workplan for future upgrades to SWMM. This workplan will address QA needs going forward including requests for new features.

6.0 Maintenance and User Support Requirements

SWMM upgrades should proceed as needed. The official EPA SWMM website will host a bug-fix list and all of the products that come out of this project that will help developers.

All source code submitted from sources within and outside of EPA will have to go through the full battery of tests to ensure the quality of the new code. However, researchers are welcome to modify and improve the code to suit their needs. It may be advantageous to others to submit any and all code modifications back to EPA for inclusion in future versions of the software.

7.0 Reports to Management

All documents related to this project will be available to management for review. The QA manager will prepare a testing review and summary every month for management review. This report will include information on all tests completed to date for this project and anticipated progress for the next month. Management may review the report and suggest changes to the testing.

References:

Browner, C. (1994) *Peer Review Program,* Washington, DC, Memorandum issued June 7, 1994.

Holden, James J. III. (1992) "Defect Analysis and TQM." *Total Quality Management for Software.* Ed. G. Gordon Schulmeyer and James I. McManus. New York: Van Nostrand Reinhold. 168-189.

McConnell, Steve. (1996). *Rabid Development: Taming Wild Software Schedules.* Microsoft Press, Redmond, Washington.

McConnell, Steve. (1998). *Software Project Survival Guide: How to Be Sure Your First Important Project isn't Your Last.* Microsoft Press, Redmond, Washington.

Reilly, W. (1993) *Peer-review Program,* Washington, DC, Memorandum issued January 19, 1993.

USEPA (2002), *SWMM Redevelopment Project Plan, Version 5.* (19 Feb. 2002).

USEPA (2000a). *Science Policy Council Handbook: Peer Review, 2nd Edition,* EPA 100-B-00-001, U.S. Environmental Protection Agency, Science Policy Council, Washington, DC, December 2000.

USEPA (2000b). *Science Policy Council: Agency Guidance for Conducting External Peer Review of Environmental Regulatory Modeling.* http://www.epa.gov/ORD/spc/modelpr.htm (12 Mar. 2001).

USEPA (1995). *Information Resources Management Policy Manual.* EPA Directive Number 2100. http://www.epa.gov/irmpoli8/polman (20 Mar. 2001).

Appendix A.
Mapping of WSWRD QA Requirements to Software and Data Management QAPP Elements

Corresponding Software and Data Management QAPP Elements	Section	WSWRD QAPP Checklist Elements
Title and Approval Sheet		Title/Signature Page
Document Control Information		
Table of Contents		Table of Contents
Distribution List		
Project Description -Background -Intended Application for Software	1.0	Problem Definition/Background
Project Description -Scope of Work -Schedule and Milestones -Planning Documents	1.1	Project Description and Schedule
Project Description -Quality Objectives for Software	1.2 2.2	Quality Objectives and Criteria for Measurement Data
Project Organization and Responsibilities	1.3	Project Organization
Functional Requirements	2.0	Experimental Design
System Design Overview Detailed Design	2.1	Sampling Method Requirements
Implementation -Development of SOPs	2.3	Analytical Methods Requirements
Implementation -QC for Implementation	3.0	Quality Control Requirements
Testing -Individual Module Tests -Integration Tests -System Testing -Regression Testing -Acceptance Testing -Beta Testing	3.1	Instrument/Equipment Testing, Inspection, and Maintenance Requirements
Project Description -Facilities Description	3.2	Instrument Calibration and Frequency
Data Validation and Verification	4.0	Data Acquisition Requirements (Non-direct Measurements) Validation and Verification Methods
Change Control and Configuration Management	4.1	Data Management
System Documentation and Archiving	4.2	Documentation and Records
Audits and Reviews	5.0	Data Review, Validation and Verification Requirements Reconciliation with Data Quality Objectives Assessments and Response Actions
Maintenance and User Support	6.0	*** ADDED ***
QA Progress Reports to Management	7.0	Reports to Management

Appendix B. Standard Operating Procedures

This project requires Standard Operating Procedures (SOP) for several processes. This appendix will have excruciating detail about the test procedures, including more detail than I can provide until I run through some mock-ups. At this time, I can provide the skeleton of the procedures.

Source Code Management SOP
To obtain a copy of the current source tree, send an email to the project QA Manager, swmmqa@epa.gov, with "get source" in the subject line.

To submit new source code to the QA Manager, attach the files to an email to the project QA Manager, swmmqa@epa.gov. The files must include complete source tree.

To enter version control, the QA Manager compares the source tree from the developer with the existing source tree. The QA Manager adds a user account to the CVS system according to the email address and submits the changes.

Testing SOP
Unit/Module Tests
1. Developer writes code.
2. Compile code against developers' source tree.

Result	Criteria	Action
Pass	Code performs to developer's expectation.	Ready for build/smoke test.
Fail	Code does not meet developer's expectation.	Further development.

Build/Smoke Tests
1. Developer submits full source tree to QA Manager.
2. QA Manager builds full implementation from developer's source tree.
3. QA Manager runs smoke files through system.

Result	Criteria	Action
Pass	No errors reported from compiler, output, or operating system.	Enter version control regression test.
Fail	Errors reported.	Generate error report.

Regression Tests

1. Iterate through the list of modified modules, compiling modified modules into the current source tree. So, at the end there will be an executable for each modified module.
2. Run each test suite on each executable.
3. Compare results with previous SWMM version.
4. Document results of tests in defect reports.
5. Write defect report to bug tracking database.
6. Submit current bug information to developers.

Result	Criteria	Action
Pass	Defects for module include previously reported defects. Defects are not significant.	Add module to current version for acceptance testing and system testing.
Fail	Defects are significant.	Generate defect report.

Acceptance Testing

1. Compile current version into executable.
2. Run test suite through executable.
3. Compare results with previous versions of SWMM.

Result	Criteria	Action
Pass	Results compare well with previous versions of SWMM.	Ready for Beta testing.
Fail	Results are dramatically different from previous versions of SWMM.	Generate defect report.

System Testing

1. Build GUI.
2. Test all components and features on the application GUI.
3. Manually run all test cases.
4. Compare results with results from application testing.

Result	Criteria	Action
Pass	All features perform required role. Results match results from application testing.	Ready for Beta testing.
Fail	Features do not perform correctly, or results do not match application testing.	Generate defect report.

Beta Testing

1. Distribute software to Technical Panel.
2. Give specific examples.
3. Request feedback.
4. Generate defect reports.

Result	Criteria	Action
Pass	Defects are not significant.	Release software.
Fail	Defects are significant.	Redevelop specific modules.

Attachment A – SWMM Project Plan