An Approach to Ensuring Quality in Environmental Software

Table of Contents for PNNL-11880:

SUMMARY

The environmental software systems developed under this approach are often used to determine impacts to the public, workers, and the environment from environmental contamination. The resulting information from systems is used in the context of important environmental decision making. It is vital, therefore, that the modeling results and the systems that provide them be scientifically defensible and capable of withstanding the most rigorous of technical reviews. In other words, the control and assurance of quality is a critical factor for environmental software systems project team (project team) in the development of environmental software systems.

This document describes the philosophy, process and activities that ensure a quality product throughout the life cycle of the development, modification, testing, and implementation of environmental software systems to analyze risk in multiple environmental media. Quality is defined as the ability of a system to meet the client's needs. Meeting client needs starts with a shared understanding of how the software must perform. It continues throughout the software life cycle through attention to details.

The environmental software systems developed by the project team are designed using an object-oriented approach. These systems offer increased benefits over those of the traditional "hard wired" systems, such as the ease of maintenance and the retention of development and testing legacy of individual components, which makes the design and testing of models and future additions faster and less costly. These systems are developed using a modular framework concept thtat allows users the flexibility to construct, combine, and couple attributes to meet their specific needs. This framework concept allows a variety of models to work within a single construct.

There are two parts to these software systems: an overall system framework and a set of modules. Each module has three components: a user interface, a scientific model, and pre/post-processors. Each of these pieces has a different set of quality criteria associated with it. However, whatever form these software systems might take for a particular client, standard processes apply to protect information from inappropriate use. These processes include application security, installation security, and protection of confidentiality, integrity, and availability of information.

The performance metrics for these software systems are grouped into eight categories: compatibility, completeness, consistency, correctness, ability to be modified, robustness, understandability, and testability. Many of the metrics in these categories are generally met in our standard approach of object-oriented design.

Figure S.1 outlines the environmental software system development process with quality check points highlighted. Although many existing models have been developed for U.S. Department of Energy, those systems may also be applicable to other agencies or organizations. Because many of our systems are designed for U.S. Environmental Protection Agency, or to be compatible with their systems, our quality process was designed to be compatible with their EPA Directive 2182, *System Design and Development Guidance* (EPA 1997). Activities roughly equivalent to their Essential Elements of Information are shown in Table S.1.

The information contained within this document can be applied to most environmental software systems developed by the project team to analyze risk in multiple environmental media, although in some cases, client needs will require an even greater level of assurance. For specific projects, clients should refer to the proposal, statement of work, and/or the project management plan for additional information on detailed quality requirements and activities being planned.

Figure S.1 Ensuring Quality in Environmental Software System Development Process

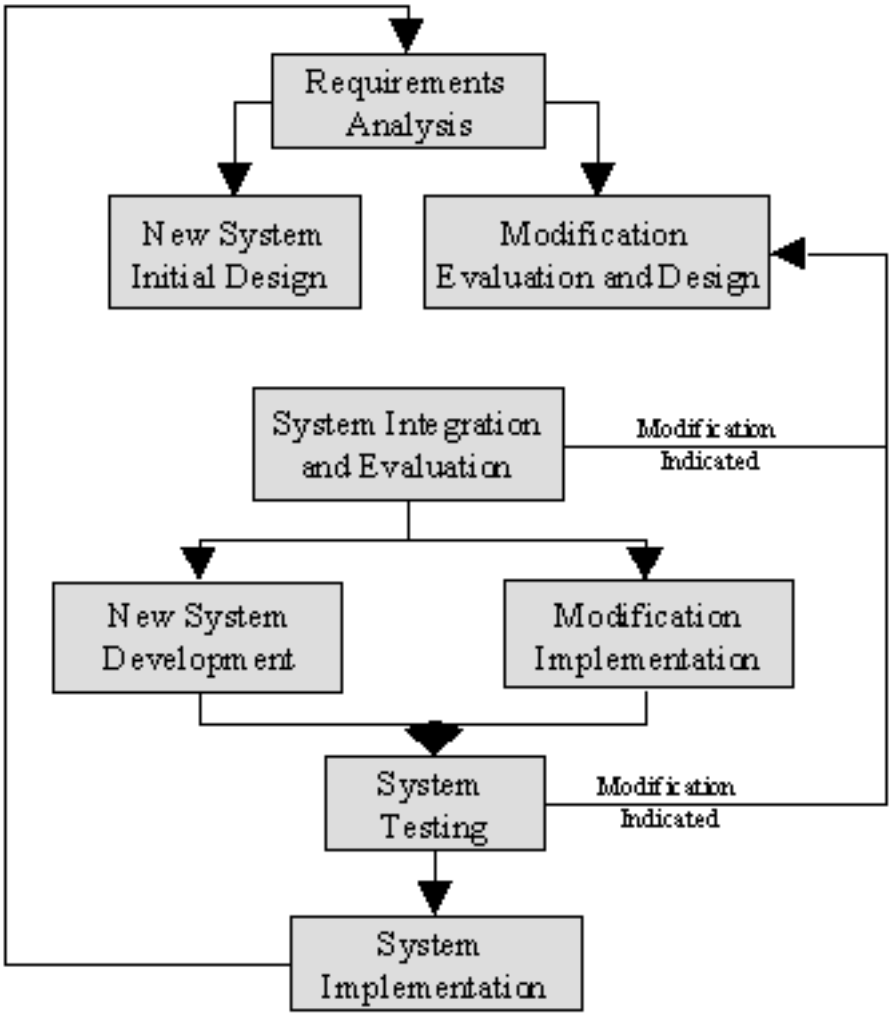

Table S.1 Relationship of Laboratory Environmental Software Development Process to U.S. Environmental Protection Agency's Essential Elements of Information (EPA 1997)

Essential Element of Information*	Environmental Software Process Equivalent (Section)
4 - System Implementation Plan	Project Management Plan or Statement of Work (3.1.2)
5 - System Detailed Requirements Document	Requirements Package (3.1.2)
6 - Software Management Plan	Project Management Plan or Statement of Work (3.1.2) and this document

7 - Software Test and Acceptance Plan	Software Test Package (5.1 and Appendix A)
8 - Software Design Document	Software Development Package (3.2.4)
9 - Software Maintenance Document	System Modification Documentation (4.2)
10 - Software Operations Document	User's Guidance and Training (3.2.3)
11 - Software User's Reference Guide	User's Guidance and Training (3.2.3)
12 - System Integration Test Reports	Software Test Package (5.1 and Appendix A)

* Elements 1 through 3 are generally completed by clients in U.S. Environmental Protection Agency before contract initiation with Pacific Northwest National Laboratory.

1.0 INTRODUCTION

A variety of environmental management regulations today require use of computer models of varying sophistication for estimating impacts of activities on humans and environment. One example is environmental remediation and restoration activities under Comprehensive Environmental Response, Compensation, and Liability Act (CERCLA) or Superfund. Another is development, implementation, and enforcement of regulations concerned with protecting human and ecological health from chemical and nonchemical human-induced contamination. These types of regulations have led to a rapidly growing need for risk analysis software systems that take a holistic approach to evaluating human health and ecological risks and hazards. Such systems assess impacts from a more comprehensive environmental systems perspective, cross-cutting various scientific disciplines. They also consider an increased number of interactions between constituents, environmental media, and receptors (Whelan et al. 1997).

Pacific Northwest National Laboratory (Laboratory), operated by Battelle for U.S. Department of Energy, has been in forefront of developing such systems for clients span federal agencies, industry, and academia. project team´s software systems are often used to determine impacts to public, workers, and environment from environmental contamination. resulting information from system is used in context of important environmental decision making, affecting not only regulatory agencies and potentially responsible parties, but decision stakeholders as well. It is vital, therefore, that the modeling results, and the systems that provide them be scientifically defensible and capable of withstanding most rigorous of technical reviews. In other words, the control and assurance of quality is a critical factor of the project team in development of software systems to analyze risk in multiple environmental media.

1.1 Purpose of This Document

This document describes the philosophy, process, and activities that ensure a quality product in the development, modification, testing, and implementation of software to analyze risk in multiple environmental media. In most cases, the process described has been used for a number of years on dozens of projects, with similar positive results. The purpose of documenting the process at this time is to:

1. Provide a ready source of information for training new project staff
2. Improve the understanding of the process and thus acceptance of the final product
3. Improve the reliability of software by ensuringthat all staff are following the same protocols
4. Improve the maintainability of software by attention to careful documentation of modifications.

The information provided should help clients, general users and project team members to understand the importance of ensuring quality in the software development life cycle.

A cornerstone of process is adherence to Laboratory standards. The Laboratory quality assurance standard states, "All staff shall document calculations, analyses, tests, and software required to substantiate results and processes used to develop products/solutions. Program managers shall manage assigned projects to a plan appropriately documents deliverables, budget, schedule, management methods, organization and control systems" (Laboratory quality assurance standard, Standards Based Management System, 1997b). In addition, the standard makes provisions for several levels of quality assurance, noting that

> *When a project meets basic Battelle requirements (as provided in Standards-Based Management System A-manuals and subject areas) or other project or activity documents that sufficiently describe how customer requirements, drivers, and business, technical, or environment, safety and health risks are met, no additional quality assurance documentation is needed*

Accordingly, this document provides the standard quality assurance planning necessary for most projects to develop, modify, evaluate, or apply software to analyze risk in multiple environmental media (some projects will require an even higher lever of assurance based on client needs).

The Laboratory also has a software development standard (Laboratory computer software and database control standard, Standards Based Management System, 1997a) that embodies the quality standard and takes several steps further. The standard requires that

> *Management shall promote utilization of recognized system life-cycle management techniques to ensure quality and repeatable delivery of information systems and infrastructure services to both internal and external customers. Staff shall take reasonable actions to safeguard the Laboratory, Battelle, and client information assets, and computing and communications applications and resources against theft, loss, misuse and disruption*.

Accordingly, this document describes how quality is managed throughout the system life cycle for environmental software systems (development, modification, testing, implementation, and application) and security measures commonly in place throughout Laboratory.

1.2 Philosophy of System Development

We define quality as the ability of system to meet client needs. Meeting client needs starts with a shared understanding of how the software must perform. It continues throughout the software life cycle through attention to details. For example, we use object-oriented programming constructs to control the flow of execution. This provides for well-defined interface points between modules and the easier maintenance of software. We also maintain a modular approach in the source program design and coding to ensure compatibility, easier testing, and clear communication points. Finally, we practice good documentation in naming conventions, symbolic parameters, paragraphing, blocking, indentation of source code, specification of a single statement per line, the intelligent use of comments and error messages so coding is easy to replicate, modify and maintain. These standard practices allow us to develop high-quality software systems that satisfy our clients.

1.3 Scope of Document

This document is meant to stand alone. The information contained within can be applied to most of the project team´s software systems to analyze risk in multiple environmental media, although in some cases client needs will require an even greater level of assurance. For specific projects, clients should refer to the proposal, statement of work, and/or project management plan for additional information on detailed quality requirements and activities being planned.

In some cases, our clients ask us to apply the models we develop to a particular problem or need (for example, in estimating risks of a major federal action for an environmental impact statement). Model application requires a different type of quality assurance process. One considers model selection, data collection, model implementation, sensitivity and uncertainty analysis, and anchoring. This type of quality assurance process is not addressed in this document.

1.4 Document Overview

The following sections provide an overview of software quality control and assurance activities associated with a software life cycle (development, modification, testing, implementation, and application). Section 2 provides a historical perspective of the development of such systems as well as a general description of software systems and certain applications and information security measures taken across projects. Section 3 describes the life-cycle process for the development of new software systems. Section 4 provides similar information for modifications to existing systems. Section 5 details test process used for both new and modified systems. Section 6 describes the support provided to transfer technology and implement the system at a client´s organization. Section 7 provides references cited elsewhere in this document. Specific guidance for quality process can be found in Appendix A. Appendix B describes roles and functions of project team members. Appendix Cprovides a glossary of specialized terms used in this document.

2.0 SOFTWARE SYSTEMS APPROACH

The environmental software systems developed by the project team are designed using an object-oriented approach. These systems offer increased benefits over those of tthe raditional "hard wired" systems. Over the past 35 years, these traditional environmental software systems have been developed for specific media (soil, groundwater, surface water, air, etc.) in an effort to understand and predict environmental phenomena. Such systems are still being developed today. The evolution of these models has followed a logical progression:

- In 1959, the Stanford Watershed Model was developed and represented as one of first "integrated" models, as it linked multiple processes by simulating the land-phase of a hydrologic cycle for an entire watershed.
- In 1969, Oak Ridge National Laboratory presented the Unified Transport Approach, which coupled (i.e., "hard-wired") detailed numerical models, describing individual environmental media. This model did not progress into general use because 1) models were difficult to understand, operate, modify, and maintain; 2) data to operate models were generally unavailable; and, most importantly, 3) computer power to drive system was lacking at time.
- In 1984, the introduction of desk-top computing allowed for the first fully coupled sequential multiple media model, which accounted for temporally and spatially varying contamination within designated media. Each medium-specific model was "hard-wired" into system, so replacing these components was not easy.
- Around 1990, the development of large multi-purpose frameworks began, which "hard-wired" a suite of codes together and investigated not just the distribution of constituents in environment but the relationships between a suite of issues deemed valuable (e.g., regulatory criteria, data quality objectives, regulatory processes, etc.).

Unfortunately, one of the drawbacks of these "hard-wired" systems was in incorporating individual components. The legacy of development and testing for the component was compromised. Even when these components could be incorporated intact, modifications were often necessary in other components. Therefore, the modification and maintenance of these systems were costly and time-consuming. A clear solution was to move toward a more object-oriented design, which is easier to maintain, retains development and testing legacy of individual components, and thus makes design and testing of models and future additions faster and less costly.

2.1 Description of Systems

That the project team develops software systems for risk analysis in multiple environmental media within a modular framework allows users the flexibility to construct, combine, and couple attributes meet their specific needs. This allows a variety of models to work within a single construct. There are two parts to these software systems: an overall system framework and a set of modules. Each module has three components: a user interface, a scientific model, and a pre/post-processor. Each of these pieces has a different set of quality criteria associated with it.

2.1.1 System Framework

The system framework typically consists of a set of modules that have been specified by a client, an associated framework user interface, and data exchange specifications. The purpose of a system framework is to:

- Minimize the data-exchange requirements between modules of framework system
- Allow relatively easy inclusion of additional modules and models
- Allow for unlimited access to data
- Address linkage concerns for a variety of models.

The system framework typically includes a user-friendly interface to enable the user to access these capabilities easily.
Depending on client needs, the framework may accommodate various levels of the detail (i.e., resolution) of models and scale of assessment (e.g., medium-specific, watershed, regional, and global). It may also access a number of site- or installation-specific and national

Each system framework is developed and maintained by a team of researchers. The team consists of at least one subject matter expert (individual responsible for communicating client needs), a framework custodian (technical expert who oversees code development for system framework), and an application expert (someone with background in application of similar systems). There may also be component developers to assist framework custodian, testers, and technical reviewers for subject matter or code. Other project team members include the project custodian (who also serves as quality assurance/quality control manager), and a documentation manager (assigned to aid in development of documentation and/or online help programs), with an overall project manager providing client interface and leadership. At the onset of specific projects, the project manager selects appropriate individuals for each of these roles. Additional information on roles and functions of each of these team members can be found in Appendix B.

2.1.2 Module

Each module potentially contains three components: the user interface, the scientific model and, for those systems that incorporate legacy models, pre-and/or post-processors. Examples of modules include source term releases, vadose zone transport, saturated zone transport, surface water transport, air transport, exposure pathway analysis, dose estimates, health impacts, and sensitivity/uncertainty support tools.

Each module is developed and maintained by a team of researchers. For each module, there will be a subject matter expert (e.g., hydrologist for groundwater module, health physicist and biochemist for dose exposure module, etc.), a module custodian (technical expert who oversees code development for a module), and an application expert (someone with background in subject area and experience applying similar models).

There may also component developers to assist module custodian, testers, a documentation manager assigned to aid in the development of documentation and/or online help programs, and technical reviewers for subject matter or code. Other project team members include the project custodian (who also serves as quality assurance/quality control manager) and task leader (generally in charge of development of each module), with an overall project manager providing client interface and leadership. At the onset of specific projects, the project manager selects appropriate individuals for each of these roles. Additional information on roles and functions of each of these team members can be found in Appendix B.

2.1.2.1 Model

A model is set of scientific calculations define a particular module. Several models have been developed over the past 10 years by researchers focusing on developing fully integrated, physics-based, intermedia modules that allow a more transparent connection between individual medium-specific models. The grouping of these physics-based models takes a holistic approach to the environmental assessment of potential constituent impacts as they simulate the following:

1. Release of constituents into environment
2. Migration and fate through various environmental media (i.e., groundwater, surface water, air, and overland surfaces)
3. Resultant exposures and impacts
4. Support tools such as sensitivity, uncertainty, graphical interface systems, and displaying results.

The overall scope of these models generally includes the evaluation of on- and off-site impacts from active and inactive sites involving both chemical and radioactive wastes. Although differing in their individual scopes, these multiple media models tend to be "analytical" in nature (e.g., mainly compartmental, analytical, semi-analytical, and empirical algorithms). Other numerical or structured-value models can be used within this holistic approach or as an outside model.

2.1.2.2 Module User Interface

The purpose of the user interface to a module is to make it easy to collect the data necessary to run model. Besides gathering necessary data, the user interface often provides online help to the user, reference storage options for collected data, flexible unit inputs, and other user support functions.

2.1.2.3 Module Pre/Post-Processors

As mentioned earlier, it saves time and cost if models can be integrated into a system framework intact. Legacy software has been tested and reviewed and can be preserved and integrated by the addition of pre- and/or post-processors

to the module. These processors transfer reorganized data into specified format of overall framework, thus allowing the inclusion of modules that were initially created for a media-specific analysis to be used in this more holistic approach to multiple media assessments. Whether a pre/post-processor is used depends on the needs of the scientific model and the specification of the framework. Models have been created or modified with specifications predefined that will likely not need pre/post-processors before integration.

2.1.3 User

The anticipated user of this overall framework and its modules is expected to have some environmental science knowledge and to be familiar with standard Windows™ application software. In addition, completion of a training session or online tutorial is also recommended for potential users. Although user interfaces are generally written to be intuitive and user-friendly, the basic understanding of the proper use and interpretation of software is recommended.

2.2 Information Security

Whatever form our software systems might take for a particular client, standard processes apply to protect information from inappropriate use. These processes include application security, installation security, and the confidentiality, integrity, and availability of information.

2.2.1 Applications Security

All computer systems and related software at Laboratory are used for official business and activities sanctioned by management. Protection systems are in place to prevent sabotage or damage by viruses. Physical security measures include following:

- Staff close and lock door to offices when systems will be left unattended for extended periods of time.
- Staff use protective measures such as screen savers and passwords.
- Staff know people who have routine access to area.
- Staff supervise use and maintenance of their systems.
- All systems are in managed buildings (access through security checkpoints).
- Approval is required before non-Laboratory personnel are granted access to computing resources.
- Backed up copies of software and files are stored in other locations (see information management below).

To protect against viruses, virus scans are performed monthly on all networked computers and at least quarterly on standalone models. In addition, all disks and files from unknown or questionable systems are scanned before use. The Laboratory uses a nationally recognized virus scanning program capable of identifying most forms of viruses including newer macro-viruses.

2.2.2 Installation Security

Most risk analysis software systems developed by the project team come in installation disk sets. Users generally install disks into the same directory or folder. To ensure ease of installation, all the software for risk analysis in multiple media is installed using general Windows installation procedures. These procedures prompt users through the installation process.

2.2.3 Information Management

All software systems covered by this approach are designed to meet client needs; each client receives specific information and software to this end. However, in all cases, master source codes remain the property of the Laboratory. Electronic copies are backed up and kept in at least two different buildings as well as on a networked fixed disk; a baseline printout is also kept separately. A completed form detailing all locations is kept by module or framework custodian for respective components as well as the project custodian. Additional guidance related to information management can be found in Appendix A.

2.3 System Safeguards and Sensitivity

These risk analysis software systems are generally used to estimate the impacts to human health from constituent releases to environment through several pathways of exposure. This information is generally used to hypothesize impacts from new constituent sources or changes to existing sources (such as effects from environmental remediation and restoration). This information can also be used to monitor compliance with environmental regulations. Clients may use it for a single activity or for analysis of a full suite of activities spanning multiple installations and locations.

Because ways in which software can be used differ between clients, the system´s sensitivity can also vary widely. Therefore, it is the client´s responsibility to determine the appropriate safeguards and security necessary for their particular use. Project staff will discuss this need with the client early in the planning process so that the client requirements can be built into the system development.

2.4 Potential Electronic Tracking System

The processes detailed in this report are currently tracked via an electronic spreadsheet program. There are, however, an increasing number of automated software quality assurance tools designed for Internet, Intranet, and web-based use that can be considered to save time and money. Change request tools introduce a consistency of communication that makes data gathering a simple, painless process and encourages people to report defects and testing time. Enhancement requests, defect reports, and changes can be easily managed from the initial incident through resolution and testing. Version control management tools can reduce the risk of change by enforcing and recording change process. The right change request and version control tools can save time, ease software maintenance, and coordinate the work of multiple team members. The project team is currently investigating the utility and feasibility of incorporating one or more of these tools to maximize quality assurance process for clients.

2.5 Performance Metrics

The performance metrics for this approach considers eight areas: compatibility, completeness, consistency, correctness, the ability to be modified, robustness, understandability, and testability. Many of these areas are generally addressed in our standard use of object-oriented design. The following are examples of performance metrics specific to four pieces of software described above (Section 2.1).

2.5.1 System Framework

Answering yes to the following questions indicates that the system framework will perform in accordance with quality expectations:

- Does the system framework design have a specification for data transfer between modules?
- Do interface requirements ensure that external modules will be compatible?
- Does the requirements package include all requirements defined in the project proposal (as documented in project management plan or statement of work)?
- Is there internal consistency between the specification requirements?
- Does the documentation use standard terminology and definitions throughout?
- Are requirements compatible with hardware and software used in operational environments?
- Do the interface requirements define required responses to potential types of errors and failure modes identified?
- Is there justification for the design/implementation constraints?
- Are requirements organized to allow for modifications?
- Are there requirements addressing fault tolerances and graceful degradation?
- Does the interface have guidance to aid the user?
- Are functional requirements in modular (object-oriented) form?
- Is formal or semiformal language used in the documentation?
- Does the documentation contain only necessary implementation details?
- Are the requirements clear and specific enough to be the basis for design guidance and functional tests?
- Does the documentation differentiate between requirements and other information provided?
- Is there a test defined for each general requirement?

2.5.2 Module User Interface

Answering yes to the following questions indicates that the module user interface will perform in accordance with quality expectations:

- Does the requirements package include all requirements defined in the project proposal (as documented in project management plan or statement of work)?
- Are the module user interface requirements consistent with framework specification requirements?
- Does the documentation use standard terminology and definitions throughout?
- Are the requirements compatible with hardware and software used in the operational environment?
- Do interface requirements define the required responses to potential types of errors and failure modes identified?
- Is there justification for design/implementation constraints?
- Are requirements organized to allow for modifications?
- Are there requirements addressing fault tolerances and graceful degradation?
- Does the interface have guidance to aid the user?
- Are functional requirements in modular (object-oriented) form?
- Is formal or semiformal language used in the documentation?
- Does the documentation contain only necessary implementation details?
- Are requirements clear and specific enough to be the basis for design guidance and functional tests?
- Does the documentation differentiate between requirements and other information provided?
- Is there a test defined for each general requirement?

2.5.3 Module Model

Answering yes to the following questions indicates that the module model will perform in accordance with quality expectations:

- Does the requirements package include all requirements defined in the project proposal (as documented in project management plan or statement of work)?
- Are formulations free of contradictions?
- Are specified algorithms and numerical techniques compatible?
- Are requirements compatible with hardware and software used in operational environment?

- Are algorithms and regulations supported by scientific or other appropriate literature?
- Is there justification for design/implementation constraints?
- Are requirements organized so as to allow for modifications?
- Are there requirements addressing fault tolerances and graceful degradation?
- Are functional requirements in modular (object-oriented) form?
- Is formal or semiformal language used in the documentation?
- Does the documentation contain only the necessary implementation details?
- Are requirements clear and specific enough to be the basis for design guidance and functional tests?
- Does the documentation differentiate between requirements and other information provided?
- Are mathematical functions defined in the documentation using notation with well-defined syntax and semantics?
- Is there a test defined for each general requirement?

2.5.4 Module Pre/Post-Processors

Answering yes to the following questions indicates that the module pre/post-processors will perform in accordance with quality expectations:

- Do processor requirements enable the compatibility of external model components to be integrated into framework system?
- Are pre/post-processors requirements consistent with framework specification requirements?
- Are requirements organized to allow for modifications?
- Are functional requirements in modular (object-oriented) form?
- Is there a test defined for processor requirements?

3.0 SYSTEM DEVELOPMENT

Software systems, developed by the project team, to analyze risk in multiple environmental media follows a routine process (Figure 3.1), honed by years of experience with a variety of clients. This development is influenced by client needs for information output, programming standards and languages, and the development tools available. In general, this process entails an analysis of client requirements, the design of the system to meet those requirements, and production and programming, including testing. The guidance discussed in the following sections can be found in Appendix A. Appendix Bdescribes the responsibilities of various roles discussed.

3.1 System Detailed Requirements Analysis

Each project starts with a definition of the client´s needs. What problem must be solved? What kinds of information are needed? Who are the ultimate users and how can the system best meet their needs? This definition begins with an analysis of needs, a definition of the functional components of hardware and software, and packaging of information.

3.1.1 Requirements Analysis

The requirements analysis is based on communication with the client and historical use of these software systems by the project team. The project team has years of experience in analyzing environmental issues and has developed significant tools for use in understanding these issues. This experience along with understanding specific needs of client is the basis for the requirements analysis, which is documented in a proposal or statement of work for the client.

An important part of the requirements analysis is to define functional components of hardware and software. These components are determined by the client´s hardware environment and the environments of legacy software being incorporated into the system.

3.1.2 Requirements Documentation

The information developed during the requirements analysis phase of project is gathered into the requirements package. This requirements description includes two levels of detail, general requirements and specific requirements. Many of the general requirements are described in project documentation, such as the project management plan with task descriptions and/or a statement of work for the project to ensure accuracy and client understanding and support. Thr statement of work is approved by the client before the initiation of work (the project management plan or statement of work contains similar information as in U.S. Environmental Protection Agency´s "System Implementation Plan" and "Software Management Plan," Essential Elements of Information 4 and 6, respectively, in EPA 1997).

In addition to the statement of work and task descriptions, additional general requirements can be documented and included into the requirements package. The requirements package should be sufficiently detailed to be used as the foundation for design and testing (the requirements package contains similar information as in EPA´s "System Detailed Requirements Document," Essential Element of Information 5 in EPA 1997).

Figure 3.1. Process for Developing Environmental Software Systems

The information in the requirements package should, at a minimum, answer the following questions:

- Which capabilities have been discussed with the client (which are they expecting the project team to use on this project)?
- What additional capabilities are necessary to produce a quality product?

- What specific restrictions have been noted?
- What potential difficulties have been identified?
- What compatibilities are necessary for usability (confirm compatibility with client´s systems)?
- Who are the project team members (subject matter expert, module or framework custodian, application expert, etc.)?

Specific requirements for a system framework include the following:

- Module data interaction specifications
- Mathematical formulations
- File formats descriptions for those files that do not meet framework system-level specifications
- Necessary help information
- Identification of ways to ensure a consistent look and feel with related interfaces
- Expected deliverables for task.

3.2 System Design and Development

The system design and development is the process of taking the information in the requirements package and translating it into software. This process is led by a module or framework custodian (see Appendix B for a full description of roles and responsibilities for this person), who may have assistance from other code developers as well as a subject matter expert. When this process has been completed, any changes to the software must be approved by the task leader and project manager.

3.2.1 Definition of Database and File Structure

Before the code is designed, the module or framework custodian (depending on component) must determine appropriate databases and file the structures of input and output from them. Module file structure should be consistent with the framework´s file specification format. Pre/post-processors may be used to aid in the conversion of legacy code file formats not already meeting those specifications. All file formats should be designed to ensure the readability and compatibility with most spreadsheet programs, and to incorporate the necessary information to communicate use and purpose of each file.

3.2.2 Code Design and Development

the code design begins with a description by the module custodians and framework custodian of major components of the design as they relate to requirements identified in the requirements analysis phase of the project. Flowcharts, block diagrams, and the relational matrix are often used to describe linkages and solution strategy. Guidance describing multiple steps involved in the design and development of software systems can be found in Appendix A.

3.2.3 Development of Software User´s Guidance

Software user guidance is developed for each module or framework. This user´s guidance, often in the form of online help, is generally an associated real-time system using hypertext language. However this type of help information is also made available for printing in hard copy form to function as a traditional user´s guide (user guidance, along with training discussed in Section 6.0, contains similar information as in EPA´s "Software Operations Document" and "Software User´s Reference Guide," Essential Elements of Information 10 and 11 in EPA 1997).

In addition, online tutorials may be developed for specified software systems. These tutorials would serve as a supplement or replacement to traditional client training arranged to aid in system implementation. These tutorials are intended to communicate the necessary information needed to operate the system correctly. Information is provided on the operation of the user interface and underlying models and their appropriate use. The development of an online tutorial is determined in the project management plan or statement of work.

3.2.4 Design and Development Documentation

The design and development activities are captured in a software development package, which identifies the type of code (new, replacement, upgrade) and members of the development team. It provides a generic description of the code, often in the form of a flowchart or task description. It also lists deliverables specific to the task such as the requirements analysis, user guidance, and testing approach. The software development package is helpful throughout the process because it captures the developers´ understanding of requirements and provides an opportunity for internal and external reviews of the design (software development package contains information similar to in EPA´s "Software Design Document," Essential Element of Information 8 in EPA 1997).

Code testing is also documented in the software development package as well as the software test package. The software development package contains a copy of the software test package and is signed by the application expert. A description of what comprises a software test package can be found in Section 5.
When completed, the software development package is reviewed by the task leader, subject matter expert, and a module or framework custodian (depending on component), and application expert. The information includes a baseline hard copy of source code listing as well as a diskette copy labeled with the component name, version, component developer names, and date. The diskette includes source codes, any executable files, and any "readme" files with special instructions to users. The package also documents the computer programming language used and any other additional languages used. Changes to components of software after the software development package is complete require the additional signature of the task leader.

Additional information for each piece of software is provided below.

3.2.5.1 Framework Design and Development Documentation

The software development package also captures a variety of information from the design phase of the project for the framework-level system, such as design requirements and communication file specifications. The software development package provides for assurance the user guidance has been developed, as well as reviews of documentation and the resolution of comments from those reviews. When the design portion of the software development package is completed, the design is approved by the subject matter expert, application expert, and the framework custodian.

3.2.5.2 Module Design and Development Documentation

The software development package also captures information from the module design phase of the project including design requirements and draft formulations needed for a module´s scientific model. Depending on the sensitivity or complexity of formulations, these formulations can be reviewed internally and/or externally, with signatures and comments noted and addressed. Reviewers can be other subject matter experts or code developers, or a representative of the client. The design portion of the software development package also addresses the need to develop pre/post-processors to enable the module to function within the overall system framework. When the design portion of software development package is completed, the module design is approved by the subject matter expert.

4.0 SYSTEM MODIFICATION

Over the years, Pacific Northwest National Laboratory has developed a variety of software systems for risk analysis in multiple environmental media such as MEPAS, RAAS, GENII, and others. While some client needs necessitate development of entirely new systems, often modifications to the existing systems are more cost-effective and useful. Modifications are influenced by programming language constraints, detailed user requirements, data requirements, and physical environment. The approach to modifications is detailed below. Descriptions of the roles and responsibilities of key project staff can be found in Appendix B. Example forms currently used in software modification tracking can be found in Appendix C.

4.1 Performance Metrics Development

As mentioned in Section 2.5, the performance metrics for our environmental software systems consider eight areas: compatibility, completeness, consistency, correctness, the ability to be modified, robustness, understandability, and testability. Many of these areas are addressed in our standard approach of object-oriented design and thus have been addressed in the development of the original systems. These systems may be modified for one of two reasons: either a client or other user has suggested an enhancement they would like to purchase or the project team or user has identified an error or "bug" in the system. The following are examples of performance metrics specific to these types of modifications.

4.1.1 Enhancements

An enhancement might be made to the framework system or one of modules. When enhancing a system framework, answering yes to following questions indicates that the system will perform in accordance with quality expectations:

- Were modifications evaluated for appropriateness and feasibility?
- Were modifications to data communication specifications documented?
- Were modifications approved by the subject area expert, applications expert, and framework custodian?
- Were modifications documented in such a way as to ensure reproducibility of the process?
- Did the baseline test cases reproduce expected results?

When enhancing a module, answering yes to following questions indicates that the module will perform in accordance with quality expectations:

- Were modifications evaluated for appropriateness and feasibility?
- Were modifications to model formulations documented?
- Were modifications approved by the subject area expert?
- Were modifications documented in such a way as to ensure reproducibility of the process?
- Did the baseline test cases reproduce expected results?

4.1.2 Errors and Bugs

Once a potential problem with the system has been identified, the project team attempts to reproduce the problem (i.e., determine the steps that led to the error message or other problem). From this, the information, module or framework custodian (depending on component) identifies a potential way to fix the problem; the proposed method is approved by the cognizant subject matter expert. When implementing the proposed change in the system, answering yes to the following questions indicates that the problem has been solved:

- Was the change evaluated for appropriateness and effect of other segments of code?
- Was the change approved by the subject area expert for a module or subject matter expert, framework custodian, and application expert for a framework?
- Was the change documented is such a way as to ensure that the steps taken can be reproduced?
- Were the baseline test cases reproduced with the expected affect on results?

4.2 System Modification Documentation

Once a software system has been developed to a baseline, modifications must be planned carefully to ensure a minimal impact on existing users. The key to this is the tracking of

requested changes and their expected impacts on results (three pieces described below - change request, change documentation, and change request summary - contain information similar to in EPA´s "Software Maintenance Document," Essential Element of Information 9 in EPA 1997).

4.2.1 Change Request

A change request may originate from a client, a project team member the client has contacted, or a project team member who has identified a need for a system modification. The process serves several purposes:

- Provides information on the potential location of the problem or enhancement
- Identifies rhe problem or enhancement
- Provides information to determine the priority of the problem or enhancement
- Documents under what circumstances the problem occurs
- Provides for concurrence between the framework or model custodian, subject matter expert, and application expert.

Sometimes circumstances prevent the suggested change from being implemented. For example, enhancements may be suggested for which a client is uninterested in paying or someone may report an error in an outdated version of code. However, the change request is documented regardless of whether the change is actually implemented. This enables staff to track issues that, while not key to improvements today, may require action later.

A change request may involve more than one problem/enhancement; in some cases, several related problems are reported at one time. The request is duplicated later in the process when each change is assigned a method of correction.

Once a change request has been initiated, it is routed through the project custodian, who will assign a tracking number, enter that number into a database, and distribute the information to the affected module or framework custodian(s). Sometimes a problem is so critical that time does not permit the physical routing of change request. In this case, the project staff send an electronic mail message to the project custodian with a brief explanation of change. This information will be entered into the database and a tracking number will be returned to the sender. Sometimes it is unclear as to which modules are involved/affected by the proposed change. In this case, the project custodian assigns a temporary tracking number until the change can be further evaluated by the module or framework custodian.

4.2.2 Change Documentation

Once a change has been documented in the change request process, the module custodian completes an evaluation of the change, which they then submit to the subject

matter expert for approval and a signature (at this point, multiple change requests that were submitted together can be distributed so each change can be tracked separately). If the change affects the system framework, an approval is based on a concurrence between the subject matter expert, framework custodian, and the application expert. The change attachment serves to document the following:

- Evaluation of the problem or enhancement by the subject matter expert(s)
- Changes made to code
- Tests run to ensure that the changes work properly
- Team members involved in the decision and change process.

If the change is denied, the module or framework custodian provides the completed change package to the project custodian for archiving. If the change is approved, the module or framework custodian begins the design and implementation of the proposed change. The application expert then tests the changes made with a baseline set of scenarios and any other testing scenarios necessary to confirm the expected affects of the change. The module custodian also provides information to explain what changes were implemented and how. Examples of such documentation includes printouts of screens, code comparisons, etc.
If a solution triggers another problem or needed enhancement, the module or framework custodian makes another change request and the process begins again.

4.2.3 Change Request Summary

When a change has been implemented, the module or framework custodian (depending on component being changed) keeps a copy of the completed change package and returns the original to the project custodian, who prepares a change request summary. This package serves to:

- Summarize information from the change request form
- Document the reviews and approvals of the subject matter experts and module or framework custodians
- Document the source code and backup updates.

The project custodian files the change request, change package, change request summary, and any associated documentation for later reference. Copies of requests originated by a client are also placed in the client folder.

4.3 Design and Development

Modification design and development follow similar guidance to the new system development process found in Section 3.2. After the initial evaluation of modification and its potential impacts to existing code and results, the system is modified and tested. Other design issues considered by project team might include the following:

- Impacts to the software user´s guidance

- Information for updating the software development package
- Potential additional reviews.

5.0 SYSTEM INTEGRATION, TESTING, AND EVALUATION

All environmental software systems developed by the project team are tested by developers before use external to Pacific Northwest National Laboratory. Each component is individually tested, and the entire system is tested to ensure compatibility. The software test package is developed by the application expert and is based on requirements for the system framework or module being tested. The test package may also be evaluated by an independent tester to ensure completeness and accurate results. Testing can often produce additional change to an already baselined system. In this case, recommended changes to the software are routed through the modification process to ensure the feasibility of those recommended changes (modification process is discussed in Section 4.0). The testing and evaluation of results is handled slightly differently for new systems verses changes to existing systems, as detailed below.

5.1 New Systems

Testing of newly developed software is key in the environmental software systems quality process (see Figure 3.1). Tests are documented in the software development package as well as the software test package. Subject area experts and module or framework custodians provide information to the application expert to summarize the reason for performing the tests and to contribute to the scope of the test. The scope information may include information on the software and documentation being tested, specific features to be tested, test case specifications, and any features that are to be excluded from testing and why. The foundation of testing is the requirements outlined in the requirements package and the development package (software test package and test results contain similar information to EPA´s "Software Test and Acceptance Plan" and "System Integration Test Reports," Essential Elements of Information 7 and 12, respectively, in EPA 1997).

Before testing, the application expert identifies major testing tasks, activities, techniques, and tools necessary to prepare for and perform testing. They also identify pass/fail criteria for each item, suspension/resumption criteria, and test deliverables. Finally, they identify the staff responsible for managing, designing, preparing, executing, and evaluating tests.

The application expert then prepares procedural steps for the test, describing such things as:

- Sequence of actions necessary for preparation before and during execution of the test
- Methods or formats for logging results of test execution and test incidents
- How test measurements will be made
- Sequence of actions necessary for shutdown, restart, execution halt, and restoration of hardware/software environment

- Actions necessary for dealing with anomalies that may occur during execution of the test.

For each test case, the module or framework custodians also note:

- Logistic Information--unique identifier assigned to test case specification, a brief description of items and features to be exercised by the test case, rationale for the selection of the test case, and pass/fail criteria for all features to be tested.
- Input/Output Specifications--all inputs and relationships among inputs required to execute the test case as well as all outputs and features required of test items.
- Requirements/Resources--characteristics and configurations of any hardware, software, or unique facility required to execute the test case as well as any constraints on the test executes test case and any dependencies between test cases.

At the conclusion of the testing, the applications expert, subject matter expert, and the module or framework custodian agree to the comprehensiveness of testing. A baseline set of test cases is documented as well as their results. These baseline test cases will be used to confirm the continued integrity of the software system when future modifications are made.

5.2 Modified Systems

For modified systems, baseline test cases are used for confirming the effects on the overall code in addition to testing needs identified in the evaluation, design, or change process. Often changes are minor enough that minimal testing is required to ensure accurate implementation. In this case, the application expert conducts the test and provides a printout to the project custodian for inclusion into the change control files discussed in Section 4.0. When the modifications are extensive, the application expert follows the same process as noted above for the testing of new systems.

5.3 General Test Scenarios

General test scenarios are prepared for new and existing software that is being introduced into this system quality approach. These scenarios are described as the minimum set of scenarios that are necessary to ensure correctness of the software produced.

For a framework system, the scenarios evaluated are based on user-friendliness and module-accessibility. Examples of the areas tested for framework systems include correct file execution, accurate file communication, and user-understandability.

For a module user interface, the scenarios evaluated are based on user-friendliness and accuracy in the data gathered for the model. Examples of the areas tested for the module user interface include the representation of model capabilities, accurate file communication, and user-understandability.

For a model, the scenarios evaluated are based on formulations and the purpose of model. Examples of areas tested for models include the correct implementation of formulations, accurate input/output data communication, and the potential combination of options.

For pre/post processors, scenarios evaluated are based on the correct data transfer and compliance with the system framework file specifications.

6.0 SYSTEM IMPLEMENTATION

The last phase of software development and ensuring quality is system implementation; however, expectations for the system implementation were first introduced in the requirements analysis phase of project. Implementation means that the specific framework developed for client can be transferred to the client to be applied at their organization by their own staff. How this system is implemented is influenced by system development, user acceptance, and operations constraints, as discussed below.

6.1 Technology Transfer

The transfer of usable software to the client is based on details depicted in a statement of work or project management plan. Client agreements are often arranged with a requirement software to be user-friendly and operable by a non-project team individual. The level of client implementation support is arranged to aid the end user of the software tool in adjusting to the look and feel of the system.

6.1.1 Client Implementation Support

Project agreements often include a client implementation support phase comprised of the delivery of software to a client and a trial time with client support through initial software usage. This support may be offered over phone, at the client location, or in training sessions on use and foundations of the software being delivered. Technical support can also be negotiated beyond the initial project agreement to enable a client to expand on software capabilities or to aid in a client´s application of software tools.

6.1.2 Implementation Documentation

Implementation documentation is usually offered as an online feature of the software system with additional documentation of file specification and formulations also available. Additional information on software user´s guidance can be found in Section 3.2.3.

6.1.3 User Training

Training on the appropriate use of the software system is recommended. This training should include information such as the capabilities of the user interfaces, reasoning for the formulations that were implemented, and limitations of the system produced. This training may occur through a formal training session, technical support of individual users, an on-line tutorial, or another form of communication capabilities.

6.2 System Operations and Maintenance

The system operations and maintenance depends on the scope of the project. In some cases, the operation and maintenance is included in the project and therefore falls into modification aspects of this document (Section 4.0). In other cases, the project ends upon delivery of the software to client; in this case, the client could negotiate continued system maintenance for future use. Guidance of the proper use and operation of the system is provided in user training

as discussed above. Documentation packages gathered for developed software systems are maintained and stored for the life of the product.

7.0 REFERENCES

EPA (U.S. Environmental Protection Agency). 1997. *System Design and Development Guidance*. EPA Directive Number 2182, U.S. Environmental Protection Agency, Washington, D.C.

Standards Based Management System. 1997a (and as updated). "Computer Software and Database Control Standard." https://sbms.pnl.gov/standard/94/9400t010.htm, Pacific Northwest National Laboratory, Richland, Washington.

Standards Based Management System. 1997b (and as updated). "Quality Assurance Planning Standard." https://sbms.pnl.gov/standard/87/8700T010.htm, Pacific Northwest National Laboratory, Richland, Washington.

Whelan, G., K. J. Castleton, J. W. Buck, G. M. Gelston, B. L. Hoopes, M. A. Pelton, D. L. Strenge, and R. N. Kickert. 1997. *Concepts of a Framework for Risk Analysis in Multimedia Environmental Systems (FRAMES).* PNNL-11748, Pacific Northwest National Laboratory, Richland, Washington.

Gelston, G.M., R.E. Lundgren, J.P. McDonald and B.L. Hoopes. May 1998. *An Approach to Ensuring Quality in Environmental Software* PNNL-11880, Pacific Northwest National Laboratory, Richland, Washington

Prepared for:

Office of Research and Development
National Environmental Research Laboratory
U.S. Environmental Protection Agency

and

Office of Environmental Management
U.S. Department of Energy

and

Radiation Protection Division
Center for Risk Modeling and Emergency Response
U.S. Environmental Protection Agency
under Contract DE-AC06-76RLO 1830

Pacific Northwest National Laboratory
Richland, Washington 99352