

U.S. DEPARTMENT OF COMMERCE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
NATIONAL WEATHER SERVICE
OFFICE OF SCIENCE AND TECHNOLOGY
METEOROLOGICAL DEVELOPMENT LABORATORY

**MDL SOFTWARE DEVELOPMENT PROCESS (SDP)
FOR THE MODEL OUTPUT STATISTICS (MOS) PROJECT**

DRAFT
February 13, 2004

TABLE OF CONTENTS

Section		Page
1.0	Introduction.....	1
1.2	Document Organization.....	1
1.3	Supporting Documents.....	2
2.0	Project Organization and Responsibilities.....	3
2.1	MOS Project Definition Overview.....	3
2.2	Other Organizations and Contractors.....	3
3.0	Project Management.....	4
3.1	Responsibilities.....	4
3.2	Management Process.....	4
3.2.1	Planning Process.....	5
3.2.2	Project Tracking and Oversight.....	6
4.0	Software Development Process.....	8
4.1	Responsibilities.....	9
4.2	Metrics.....	10
4.3	Detailed Software Process Activities.....	11
4.3.1	Requirement Analysis.....	11
4.3.2	Design.....	12
4.3.3	Code and Unit Testing.....	14
4.3.4	Software Integration Testing.....	15
4.3.5	System Integration Testing.....	17
5.0	Documentation.....	18
5.1	Formal Documentation.....	18
5.2	Informal Documentation.....	20
5.3	Software Development Files.....	20
6.0	Reviews.....	21
6.1	Peer Reviews.....	21
6.2	Code Walkthroughs.....	22
6.3	Management Reviews.....	22
6.3.1	Project Status Reviews.....	22
6.3.2	Development Reviews.....	22
7.0	Testing.....	24
8.0	Configuration Management.....	25
9.0	Software Standards.....	26
10.0	Development and Testing Environment.....	27
11.0	References.....	27

TABLE OF CONTENTS (continued)

Figures

Figure 4.0-1	Data Flow of Software Development Process.....	8
Figure 4.1-1	Development Organization.....	9

Tables

Table 3.2.1-1	Planning Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	6
Table 4.3.1-1	Requirements Analysis Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	12
Table 4.3.2-1	Design Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	13
Table 4.3.3-1	Code and Unit Testing Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	15
Table 4.3.4-1	SwIT Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	16
Table 4.3.5-1	System Integration Testing Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	17

**MDL SOFTWARE DEVELOPMENT PROCESS FOR THE
MODEL OUTPUT STATISTICS (MOS) PROJECT**

1.0 INTRODUCTION

The Meteorological Development Laboratory (MDL) Software Development Process for the Model Output Statistics (MOS) project establishes the software and development processes that is used throughout the software development life cycle. This document describes the processes and procedures that are used to design, implement, and test software.

1.1 DOCUMENT ORGANIZATION

The SDP is organized into the following ten sections.

Section 1 - The Introduction presents the purpose and scope of the plan, an overview of the project, and related documentation.

Section 2 - The Project Organization and Responsibilities section discusses the project organization and the responsibilities for the MOS project and other organizations that interface with the project during the development process.

Section 3 - The Project Management section describes a standard approach and mechanism for project managers to plan, track, and measure the software development process.

Section 4 - The Software Development Process section presents an overview of the software development life cycle and describes the major activities in each life cycle phase. It establishes the software development process, methods and standards to be used in the development of MOS software.

Section 5 - Documentation describes documentation created and the method for the retention of that documentation

Section 6 - Reviews describes the types of reviews, what is being reviewed, when reviews are employed and policies and procedures associated with each review

Section 7 - Testing explains the concept and activities employed in testing MOS software.

Section 8 - The Configuration Management section describes the concepts and activities used for the management of the software development and test products.

Section 9 - Software Standards identifies the standards pertaining to software development including coding standards and design rules.

Section 10 - The Environment section describes the hardware and software environments used to develop and test MOS project software.

1.2 SUPPORTING DOCUMENTS

Other important documents related to this document include:

MDL Standards, Guidelines and Procedures (NWS 2003a) - This document contains the software standards used in the development of MOS software and contains the set of procedures and guidelines to be used by the development team to standardize the development process and ensure that it is a repeatable process.

2.0 PROJECT ORGANIZATION AND RESPONSIBILITIES

This section describes the responsibilities within the MOS project and the Government and Contractor organizations that interface with the project during the development of application software.

2.1 MOS PROJECT DEFINITION OVERVIEW

The objective of the MOS project is to develop objective guidance for weather elements contained in public, aviation, marine, fire weather, and hydrological service forecasts for projections of 3 hours to 14 days in advance. To develop new techniques for generating objective guidance with emphasis on probabilistic and high-resolution gridded guidance. To generate and disseminate objective guidance products. The MOS project consists of three tasks:

Statistical Forecast Development - Develop and improve objective techniques for predicting weather elements needed in official NWS forecasts for projections ranging from 3 hours to 14 days in advance.

Advanced Meteorological Applications - Investigate and apply scientific approaches to produce interpretive guidance on high resolution grids and from ensembles, and to apply advanced Geographical Information System and statistical techniques to the output of objective forecast models.

Operations and Software Support - Design, evolve, implement, and maintain systems to produce, disseminate, and archive objective guidance.

2.2 OTHER ORGANIZATIONS AND CONTRACTORS

MDL staff must interface regularly with other organizations inside and outside of MDL that play a critical role in the development and deployment of MOS software. These organizations include:

RS Information Systems (RSIS) - RSIS is responsible for managing and staffing MDL's system administration group and for providing software developers who participate in MOS development.

OST/MDL/Product Generation Branch (PGB) - PGB is responsible for developing and implementing software that ingests and displays MOS products on an AWIPS platform.

Office of Science and Technology (OST) - OST has overall management responsibility for the MDL programs.

National Centers for Environmental Prediction (NCEP) - NCEP manages the operational computer system and is responsible for implementing new products to the MOS forecast package.

3.0 PROJECT MANAGEMENT

This section describes a standard approach and mechanism for project managers to plan, track, and measure the software development process.

3.1 RESPONSIBILITIES

The responsibilities of project management are described below.

Director, MDL - Overall responsibility for the software development effort, initiates change requests and is the Software Engineer (SE) for MOS software development.

Project Manager - The Project Manager is responsible for reviewing the plans, making the project commitments, and reviewing any changes. Specifically, the Project Manager oversees cost, schedule, and interfaces with other NWS organizations. The Project Manager is responsible for initiating change request, defining tasks, participating in all requirement, design, and code walkthroughs, and works with the Library Team (LT). The Project Manager meets with the LT every other week to present proposed changes for approval and/or further action. The Project Manager conducts regular oversight reviews with the MDL Director and MDL Deputy Director.

Task Manager - Statistical Forecast Development - The Statistical Forecast Development Lead is responsible for overseeing the development and improvement of objective techniques for predicting weather elements needed in official NWS forecasts for projections ranging from 3 hours to 14 days in advance.

Task Manager - Advanced Statistical Applications - The Advanced Statistical Applications Lead is responsible for applying advanced statistical forecast techniques to the output of objective forecast models, including model ensembles or MOS guidance, to enhance the prediction of weather elements needed in official NWS forecasts. Emphasis is on improving probabilistic forecasts for projections ranging from 1 to 14 days in advance.

Task Manager - Operations and Software Support - The Operations and Software Support Lead is responsible for the design, implementation, and maintenance of systems used to produce, disseminate, and archive objective guidance.

3.2 MANAGEMENT PROCESS

Managing a software project requires careful planning, control of activities, and tracking against the planned activities. Once a plan is developed, the actual activities are tracked against the plan to determine whether there is any deviation from the plan. Metrics (as defined in Section 4.2) are used to measure performance and suggest process improvement.

This management process will mature as the program progresses. This means that the plans are living documents. This management process emphasizes early planning and risk analysis. The plans are reviewed, revised, and expanded based on the most recent knowledge of the program at key milestones, when the scope changes, and at regular intervals.

3.2.1 PLANNING

Software planning involves developing estimates for the work to be done, establishing the necessary commitments, and defining a plan to complete the work. The plan provides the basis for initiating the software effort and managing the work. Accurate estimations of cost and schedule up front and adherence to required staffing levels and equipment usage are a key factor to completing a project within budget.

Planning is a continuous process. As the design proceeds, certain design decisions may change the plan or schedule. As change requests (e.g., new deficiencies, enhancements, or requirements) are identified, the process must be repeated and documentation produced describing the impact of any changes to the cost and schedule.

The following functions are required during the Planning step. The responsible party for each function is shown in **bold** type.

Prepare MOS Change Requests - MOS Change Request (MCRs) can be prepared by **MDL Director, Project Manager, or NCEP** for new requirements, enhancements, hardware and system changes. For developmental software, these change requests are usually established by the needs of the developer; operational software is bound by the constraints of the NCEP operational environment.

Create Tasks - Tasks are prepared by **Project Manager** for new requirements, enhancements, hardware and system changes. Tasks are linked to MCRs.

Estimate Level of Effort (ELOE) - For each task the **Project Manager** estimates the size of the task and the ELOE. The size of the task includes the number of software components required and lines of code. The ELOE is measured by estimating the number of labor hours required to develop, test, document, and maintain the software.

Prepare Project Tracking Information (PTI) - The **Project Manager** prepares a Staffing Plan and a Development Schedule.

Table 3.2.1-1 describes the input, output, entrance and exit criteria, process controls and metrics for this Planning step along with the responsible parties.

Table 3.2.1-1. Planning Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	MOS Change Request Form received	Project Manager, MDL Director, NCEP
Input	MOS Change Request Form prepared Other development schedules	Project Manager, MDL Director, NCEP
Output	Project Tracking Information (PTI)	Project Manager
Exit Criteria	PTI completed	Project Manager
Process Controls	Management review	MDL Director
Metrics	None	

3.2.2 TRACKING AND OVERSIGHT

Software tracking and oversight involve tracking and reviewing software progress against the documented estimates, schedules, and plans, and adjusting these based upon the actual data. This activity occurs throughout the development effort.

Software tracking and oversight start as soon as the effort commences. The development schedules are used as the basis for tracking and oversight throughout the development life cycle. During development, actual data is collected according to the process model. These data are analyzed at specified intervals against the plan. If the actual status of the program deviates beyond an acceptable norm, corrective action is taken. Corrections made to schedule, cost, or software sizes are reviewed with respect to each other. Then, the schedule and any renegotiated commitments are revised and reviewed.

When modifying the schedule, the Software Manager (SM) keeps records that explain the reasons for various corrective actions and the rationale for that change. This information is useful for developing the lessons learned and other postmortem analysis.

The following are reviews that are established to track development schedule.

Status Reviews - Status reviews, if necessary, are conducted with the Project Manager and developer to review progress and address issues. These can be conducted weekly or biweekly depending on the wishes of the Project Manager.

Planned Reviews - Planned reviews are conducted at key periods during the development cycle to review requirements, design, coding, testing, and delivery. In addition, reviews are conducted to review code (e.g., code walkthrough) and other development products (e.g., peer reviews). See Reviews (Section 6.0) for more information on these reviews.

MDL Staff Meeting - The MDL staff meeting is conducted weekly to coordinate between MDL Director, Project Manager and Software Manager.

In addition, following information is used by the Project Manager to provide tracking and oversight, and facilitate reporting to other NWS organizations.

Project Tracking Information (PTI)

- Development Schedule - High level schedule showing the development activities (e.g., requirement review) and major checkpoints on a time line.
- Staffing Plan - List of tasks per release with estimated level of effort and identified Task Manager, Task Lead and development staff.
- Development Checklist - Internal detailed checklist employed by the management to track development progress and adherence to the MDL Software Development Approach requirements (NWS 2004a).

See the MDL Standards, Guidelines, and Procedures (NWS2003a) for a PTI template.

Metrics

- Information to guide process improvement (See Section 4.2).

4.0 SOFTWARE DEVELOPMENT PROCESS

This section describes the software development and maintenance process in detail. Below are the major activities that should be accomplished during the development of software.

- Requirements Analysis
- Design
- Coding and Unit Testing
- Software Integration Testing
- System Integration Testing

Some of these activities may not be necessary for specific types of projects. The process is flexible and can be ordered to fit any of the popular paradigms of software architecture including Waterfall, Rapid Prototype, or Evolutionary models. Figure 4.0-1 shows the typical waterfall software development process sequence with interrelationships, and significant checkpoints and reviews.

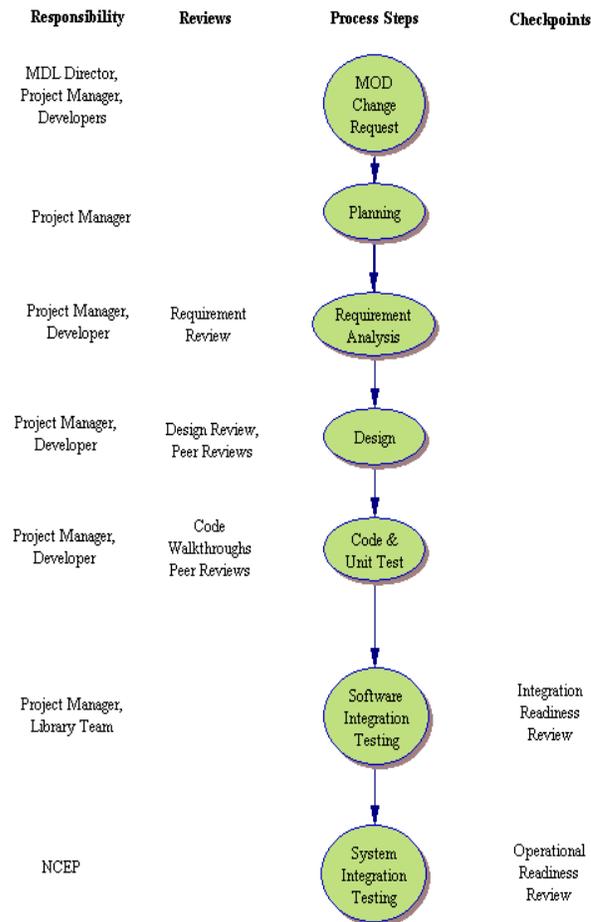


Figure 4.0-1: Data flow of the Software Development Process.

4.1 RESPONSIBILITIES

The development organization is shown in Figure 4.1-1 and defined below. Depending on the size of the development effort, a staff member may be tasked with multiple roles.

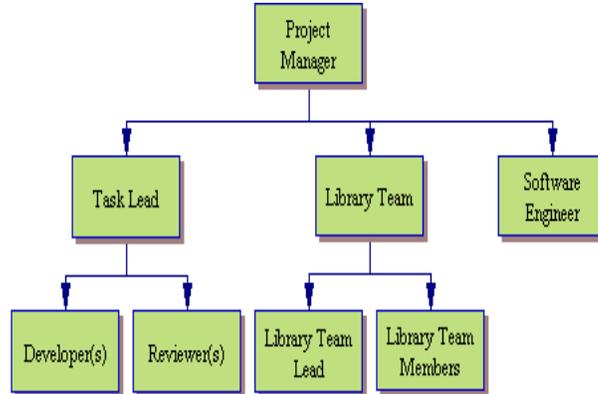


Figure 4.1-1. Development Organization

Task Lead - The Task Lead's responsibilities leads the development of the task. The Task Lead coordinate the activities of the developers and reviewers. The Task Lead performs the requirement analysis, assists with the design, reviews developers code and test procedures, and assists the preparation of test plans and schedules. The Task Lead reports to the Project Manager.

Library Team - The Library Team consisting of four (4) to six (6) members is responsible for monitoring and testing proposed changes. The Library Team meets with the Project Manager to present those proposed changes for approval and/or further action.

Software Engineer - The Software Engineer is the chief architect and approves all design and code walkthroughs.

Developer - The developer's major responsibility is to code software using the established coding standards, guidelines, and procedures. After completing the coding phase, the developers go through the code review process with their Project Manager. The developer is responsible for creating test procedures, performing the unit testing, and submitting a Development Ticket to the Library Team when coding is completed and the software is ready for integration and final testing. The developer is responsible for preparing all necessary external documentation.

Reviewers - The Reviewers role is to participate in the various reviews described in Section 6.0 of this document. The reviewers can be the Project Manager, Software Engineer, technical staff members, developers, specialists, or users whose backgrounds give them insight into the material to be discussed.

Library Team Leader - The Library Team (LT) Leader assigns incoming Development Tickets to one or more members. Prior to each LT meeting, the LT Leader review any changes made by the Software Engineer. The LT Leader schedules meetings of the LT.

Library Team Member - Library Team (LT) Member will be assigned incoming Development Ticket by the LT Leader. The LT Member is responsibility for assessing the significance of revised codes, verifying that all codes have gone through the MOS project software process, and running pre-established test cases with the updated libraries to verify functionality and performance.

4.2 METRICS

The Project/Software Managers use software metrics gathered to evaluate key characteristics of the software being developed, the process employed, and the associated management indicators of progress. A successful metrics program depends on accurate and consistent data collection and presentation. The validity of the data should be determined prior to any analysis activity. Under these circumstances, the metrics can provide early warning of potential software development problems. In turn, this should lead to early problem resolution.

Graphic presentation of the metrics can reveal developing trends, which, when analyzed as related sets, highlight anomalies that might otherwise be overlooked. Management can then determine their significance and corrective action can be taken. Results are used to improve the ongoing project and are reported at management reviews.

Types of metrics include number of software requirements, number of software requirement changes, product size, level of effort, cost, schedule, defects, and computer resource utilization. In the future, MDL will develop composite metrics to indicate productivity, quality, production rate, and stability.

Metrics used by the MOS project are identified in Section 4.3 of this document.

4.3 DETAILED SOFTWARE PROCESS ACTIVITIES

For each step in the process, this document defines:

Entrance Criteria - criteria needed to start the activity,
Input - products necessary for this step,
Functions - process and tasks of each step,
Metrics - set of measurement data resulting from the work products,
Responsibility - who is responsible for completing that activity,
Output - products of activity,
Exit criteria - criteria for completing the activity, and
Process controls - controls put into place to insure quality.

Guidelines and process descriptions necessary to complete the activity are kept in the MDL Standards, Guidelines and Procedures (NWS 2003a).

4.3.1 REQUIREMENTS ANALYSIS

The software development processes are requirements driven. Requirements are a formal statement of an attribute to be possessed by the product or a function to be performed by the product. These requirements form an agreement between developer and customer. Requirement analysis provides a means for establishing and maintaining requirements so that both the customer and the developers are working from the same set of expectations.

The completion of the Planning (Section 3.2.1) activity results in a task(s) being assigned to a Task Lead or Developer. The Task Lead or Developer defines and develops software requirements and prepares a Requirements Description (RD). The Requirement Description is a set of detailed software requirements derived from the MOS Change Request (MRC). Requirements can address operation concepts, functional and user interface specifications, performance and capability, external interfaces, security, error handling, installation, configuration, language, maintenance, and the use of legacy software. For data-driven and data-intensive systems, the Requirement Description can include data sources, types, and rates.

A Requirement Review is held to finalize the understanding of requirements with the Project Manager and obtain approval.

Note that requirements can be dynamic and change as the design is evaluated and development is conducted. As these changes occur, the changes should be made to the Requirements Description and those changes presented at the appropriate review.

The following functions are required during the Requirements Analysis step. The responsible party for each function is shown in **bold** type.

Define and develop software requirements - The **Task Lead or Developer** defines the set of software and derived requirements for this project and prepares a Requirement Description.

Analyze requirements - A peer review is conducted by the **Task Lead or Developer** to analyze the requirements to ensure, at a minimum, traceability, completeness, clarity, testability, safety, and validity.

Perform Requirements Review - A Requirements Review is conducted by the **Task Lead or Developer**. Participants should include **Project Manager**. Action items (if necessary) are documented and delivered to the approval body and tracked through closure by the **Task Lead or Developer**.

Approval - The Requirements Review is formally approved by **Project Manager**. The approval can be provided verbally, in an email, or memo.

Update Project Planning Information (PTI) - The **Project Manager** updates the Development Schedule and Staffing Plan with any changes that result from a better understanding of the requirements.

Table 4.3.1-1 describes the input, output, entrance and exit criteria, process controls and metrics for this Requirements Analysis step along with the responsible parties.

Table 4.3.1-1. Requirements Analysis Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Project Tracking Information (PTI) prepared	Project Manager
Input	MOS Change Request	MDL Director, Project Manager, NCEP
Output	Requirements Description	Task Lead, Developer
Exit Criteria	Requirements Description approval	Project Manager
Process Controls	Requirements Review Management Approval	Task Lead, Developer
Metrics	To Be Determined (TBD)	

4.3.2 DESIGN

The Design phase will establish a complete software design to be used by the developers of the software. The software components are defined in terms of purpose, use cases, interfaces, data requirements, data structure, error handling, storage and throughput, timing requirements, and diagnostic considerations. The relationship of components is defined in terms of data flow between them and external interfaces, and the control flow between components.

A Design Review is held to finalize the understanding of design with the system engineering community and obtain approval to code the software. Approval of the design is provided by the Project Manager and Software Engineer.

The following functions are required during the Design step. The responsible party for each function is shown in **bold** type.

Define software components - The **Task Lead or Developer** defines the set of software components to be developed or modified.

Prepare design - The **Task Lead or Developer** defines components and interfaces, relationships and data flows, physical structure, user interface, data structure, and critical test scenarios.

Peer Review - Peer reviews are conducted by the **Task Lead or Developer** to review selected design components. Participants could include the Task Lead and other Developers.

Perform Design Review - A Design Review is held to understand a basic design when new software development is undertaken for either developmental or operational codes. No strict process is followed. The review process depends on the code that needs to be written. If a developmental code that needs to be completed does not deviate much from a previous code, the review process is very informal. A more extensive review is undertaken in instances when a code departs extensively from previously written software.

A Design review is conducted by the **Task Lead or Developer**. Participants should include Project Manager and Software Engineer. Action items (if necessary) are documented and delivered to the approval body and tracked through closure by the **Task Lead or Developer**.

Approval - The Design Review is formally approved by the **Software Engineer and Project Manager**. The approval can be provided verbally, in an email, or memo.

Assign developers - The **Project Manager** assigns developers to the software components.

Update Project Tracking Information (PTI) - The **Project Manager** updates the Development Schedule and Staffing Plan.

Table 4.3.2-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this Design step along with the responsible parties.

Table 4.3.2-1. Design Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Requirements Description Approval	Project Manager
Input	Requirements Description	Task Lead, Developer
Output	Design documents	Task Lead, Developer
	Updated Development Schedule	Project Manager
Exit Criteria	Design completed and approved	Project Manager, Software Engineer
Process Control	Design Review	Project Manager, Software Engineer
Metrics	TBD	

4.3.3 CODE AND UNIT TESTING

Once the Design Review is approved, the developers assigned to the Coding activity can begin.

Coding is performed uniformly across software products using defined standards and guidelines. The objectives of source code written are as follows:

- Meets requirements,
- Contains correct logic and interfaces and handles data structures, properly as specified in the design documentation,
- Complies with coding standards,
- Compiles successfully without any warning or error messages,
- Follows good coding techniques,
- Includes proper internal documentation,
- Follows reasonable and understandable size limitations, and
- Considers reuse, portability, and system independence.

The following functions are required during the Code and Unit Testing step. The responsible party for each function is shown in **bold** type.

Prepare code - The code is created or modified by the **Developer** according to the design documents and MDL Standards, Guidelines and Procedures (NWS 2003a).

Create Test Procedures - Test procedures are created by the **Developer** to test the modification to the code. Additional test procedures are identified to ensure the modification has not affected the existing code. Test Procedures should be prepared in accordance with guidance contained in the MDL Standards, Guidelines and Procedures (NWS 2003a).

Test Procedure Peer Review - The **Developer** should conduct a peer review of the test procedures.

Perform Unit Testing - The **Developer** should perform unit testing in accordance with MDL Standards, Guidelines and Procedures (NWS 2003a). In most instances, the software developer is responsible for checkout of the software to ensure functionality. It is the responsibility of the developer to be sure that answers are correctly obtained by comparing the answers of older codes. The **Project Manager** supervises the testing.

Perform Code Walkthroughs - Once the code has been completed and Unit testing has been completed, the **Developer** prepares and performs a Code Walkthrough. Participants in the code walkthrough include the Project Manager and Software Engineer. This is typically an informal process, however, a Code Walkthrough form needs to be filled out and defects recorded.

Fix Defects - The **Developer** fixes defects found during the Code Walkthrough.

Prepare Draft MDL Office Note - The **Developer** prepares a first draft of the formal program documentation which is usually incorporated into a MDL Office Note.

Submit Development Ticket (DT) - The **Developer** submits a Development Ticket to the MOS Library Management System. Development Tickets are used for any and all changes so that there is a permanent record of all changes. Each ticket is also printed out and stored as a hard or backup copy.

Table 4.3.3-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this Code and Unit Testing step along with the responsible parties.

Table 4.3.3-1. Code and Unit Testing Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Design Completed	Task Lead, Developer
Input	Design Document	Task Lead, Developer
Output	Code, Test procedures, Code Walkthrough documentation, Draft Office Note, Development Ticket	Developer
Exit Criteria	Code is tested, walkthrough is completed, defects are addressed, draft office note is completed	Developer
Process Controls	Code Walkthrough Peer Reviews	Developer Developer
Metrics	TBD	

4.3.4 SOFTWARE INTEGRATION TESTING (SwIT)

The MOS project established an internal process to test software in an integrated environment and maintain developmental software libraries. A rotating Library Team consisting of 4 to 6 people is responsible for monitoring and testing Development Tickets submitted by the developers. The Library Team meets every other week with the Project Manager to present those proposed changes for approval and/or further action.

The following functions are required during the SwIT step. The responsible party for each function is shown in **bold** type.

Assign a Development Ticket to a Library Team Member - The **Library Team Leader** assigns Development Tickets to a member of the library team.

Assess Code Changes - The **Library Team Member** assesses the code change and insures that the code has gone through each step of the MOS software development process (e.g., design review, code walkthrough)

Run Test Procedures - The **Developers** run the assigned test procedures for the Development Ticket and record results. The process steps are as follows:

- Make a test version of the library to be changed. Copy all pieces of the current library to a working directory.
- Copy the new or modified codes for that library to a test library (/nfsuser/g06/mos2k/libteam/test on the IBM).
- Make a new linkable of that library. Most libraries have a script called make__lib.sh. Use this to create the linkables.

- Test the new library
 - Create new executables for any code that will be affected by the changes.
 - If an enhancement or error modification is made, test each situation using the current executable for that code, and a new executable that has been created with the Library Team member's new library.
- Update the mos2k library on the IBM
 - Copy the updated version of the existing code into the mos2k library (/nfsuser/g06/mos2k). Make notations in the README file of all new and modified codes that were added to the library. Run the make_lib.sh script to update the linkables. This script works by compiling all .f codes.

Conduct an Integration Readiness Review (IRR) - The **Library Team** meets every other week with the Project Manager to present those Development Tickets to determine the readiness for System Integration Testing. Action items (if necessary) are documented.

When the proposed changes are approved, the Library Team prepares a report for the Project Manager to outline any changes made, and also tells the Project Manager and operations team of any changes that must be moved back to the HP and operational libraries, respectively.

Approval - The IRR is formally approved by the **Project Manager and Software Engineer**. The approval can be provided verbally, in an email, or memo.

Update libraries for Software Engineer - The **Project Manager** will move all modified or new codes from the IBM to the Software Engineer's working environment on the workstations after notification by the Library Team.

Table 4.3.4-1 describes the input, output, entrance and exit criteria, process controls, and metrics for this SwIT step along with the responsible parties.

Table 4.3.4-1. SwIT Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Development ticket received and assigned to library team	Developer
Input	Development Ticket	Developer
Output	Tests are completed and checked out	Library team member
Exit Criteria	Testing completed, Integration Readiness review conducted, and software is approved	Library team
	Library Updated	Project Manager
Process Control	Integration Readiness Review	Library team
Metrics	TBD	

4.3.5 SYSTEM INTEGRATION TESTING (SIT)

System Integration Testing (SIT) takes place in the NCEP environment. When software additions or changes are ready to be implemented in the NCEP operational environment, a member of the MOS team completes an NCEP Change Request Form. This CR describes what additions or changes were made so that NCEP understands what changes they are implementing. NCEP will run the modified code and notify the MOS team. The MOS team reviews results of the test runs and makes sure the information is correct. If the test fails, the MOS team notifies NCEP and another test is arranged.

The following functions are required during the SIT step. The responsible party for each function is shown in **bold** type.

Prepare NCEP Change Request - The **Operations Task Lead** creates an NCEP Change Request Form.

NCEP Runs Test Procedure - **NCEP** runs the assigned test procedures for the NCEP Change Request, records the results, and provides MOS project management with those results.

Operational Readiness Review (ORR) - The **Operations Task Lead** will review, approve or reject the test. If the test fails, another test is arranged with NCEP.

Prepare Final MDL Office Note - The **Developer** prepares a final of the MDL Office Note.

Table 4.3.5-1 describes the input, output, entrance and exit criteria, process control, and metrics for this SIT step along with the responsible parties.

Table 4.3.5-1. System Integration Testing Input, Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	Integration Readiness review conducted and software is approved	Library team
Input	Completed software and draft MDL Office Note	Operations Task Lead
Output	NCEP Test Results, Final Office Note	NCEP, Developer
Exit Criteria	Successful ORR	Operations Task Lead
Process Control	Operational Readiness Review	Operations Task Lead
Metrics	TBD	

5.0 DOCUMENTATION

Documentation is continually prepared to communicate and archive valuable development information. This information is used by management, system integrators, users, developers, and support and maintenance personnel. All documentation is updated periodically and archived for all software development in the Software Development Files (SDFs).

5.1 FORMAL DOCUMENTATION

This section identifies what formal external documentation is produced for software developed by the MOS project. Internal documentation requirements and standards are covered under the appropriate software standards described in Section 9.0 of this document.

Requirements Description - Description of the requirements is based on the MCR. Guidelines for preparing requirements and a checklist are contained in the MDL Standards, Guidelines and Procedures (NWS 2003a).

Design Document - Documentation used by the MOS project to capture the design of an application. Guidelines for preparing a design and a checklist are contained in the MDL Standards, Guidelines and Procedures (NWS 2003a).

Project Tracking Information (PTI) - See the MDL Standards, Guidelines and Procedures (NWS 2003a) for a PTI template.

- Development Schedule - High level schedule showing the development activities (e.g., requirement review) and major checkpoints on a time line.
- Staffing Plan - List of tasks per release with estimated level of effort and identified Task Manager, Task Lead and development staff.
- Development Checklist - Internal detailed checklist employed by the management to track development progress and adherence to the MDL Software Development Approach requirements (NWS 2004a).

Test Plan - Document describing testing to be performed on an application. This plan includes focus of testing, listing of test procedures for the by test categories, test environment, test tools, and schedule and staffing. Guidelines for preparing a test plan are contained in the MDL Standards, Guidelines and Procedures (NWS 2003a).

Test Procedures - One or more test procedures are identified to evaluate the functional or structural condition of the code. Test procedures are designed based on specific functional requirements or components of code structure. Each test procedure will identify the software requirements validated by the test. Guidelines for preparing test plans and test procedures are contained in the MDL Standards, Guidelines and Procedures (NWS 2003a).

MOS Change Request (MCR) - The MCRs are prepared for new requirements, enhancements, hardware, and system changes. For developmental software, these change requests are usually established by the needs of the developer; operational software is bound by the constraints of the NCEP operational environment. The MCR includes the following information:

- Tracking number
- Purpose
- Originator (name and title)
- Title of change
- Description of change
- Type of change (hardware, software, documentation)
- Date submitted
- Required change date including rationale

MOS Development Ticket (MDT) - The MOS Development Ticket documents a software change to be reviewed by the Library Team. The Development Ticket includes the following information:

- Ticket number
- Submitted by
- Name of code
- Library
- Description
- Dependencies
- Priority
- Location of code and test data
- Verification that each step in the development and testing process was completed (e.g., Requirement review, design review, code walkthrough,, test procedures, documentation).

NCEP Change Request (NCR) - The NCR is prepared by MOS project personnel to install operational software on NCEP's IBM. The NCR include the following information:

- Date submitted for approval
- Information about the person approving and submitting the NCR
 - name, phone number, title, routing code, email address
- NCEP personnel assigned the NCR
- Registration number
- Date scheduled to be implemented
- Memo Date
- Production program Name
- Production Job Name(s)
- Package Name
- Location of:
 - Source
 - Job/Task Script(s)
 - Model (USH) Script(s)
 - Parameter Fields
 - Fixed Fields
- Purpose of Program
- Description of Change
- Implementation Instructions

Office Notes - Modification or addition to the MDL Office Note 00-02 is the formal documentation prepared for the software routines that are part of the MOS-2000 system. See the MDL Standards, Guidelines and Procedures (NWS 2003a) for more information concerning the content and style of the Office Note.

5.2 INFORMAL DOCUMENTATION

Informal documentation includes:

- Briefing slides,
- Design information and documentation, including rationale supporting design decisions,
- Peer review, design review, and code walkthrough results, checklists, and comments,
- Test results, and
- Meeting minutes, memos, action items, checklists, and correspondences.

5.3 SOFTWARE DEVELOPMENT FILES

Software Development Files (SDFs) contain all formal and informal information describing the development or maintenance of the software product. The SDF should be maintained online as much as possible, to facilitate searching and inclusion into documentation. It may contain media copies or references to controlled media copies of supporting data.

6.0 REVIEWS

This section defines the process for the five types of internal reviews that are performed during the development of software. These reviews are categorized as:

- Peer reviews
- Code walkthroughs
- Project Status Reviews
- Development Reviews

Some of these reviews are conducted internal to the MOS project and, in most cases, do not include customer participation.

Our experience has proven the effectiveness of internal reviews throughout the software development and maintenance process. This is an extremely cost-effective approach for early identification and resolution of technical and management problems and improved communication within the software project team. Furthermore, the types of reviews defined in this section work equally well on all sizes and types of software projects. However, each type of review must be exercised in an appropriate manner, as defined in this section, or substantial benefits may be degraded.

The Software Manager responsibility includes participation in selected reviews, usually as an observer, and verification that the reviews are taking place in conformance with the process. This responsibility includes periodic auditing of selected SDFs to ensure that the material is updated and current.

6.1 PEER REVIEWS

Peer reviews are conducted according to a documented procedure. Staff is trained in peer review objectives, principles, and methods from the perspective of both leader and participant.

The concept behind peer reviews is that the author/developer of a product (i.e., specification, design, unit of code, test procedures) gets help from a colleague who is familiar with the product. Together, they discuss in detail a specific portion of the overall product. The author presents the product element to the colleague, item by item, who in turn raises questions and suggestions. Application of the concept is simple and inexpensive. One-on-one reviews are ad hoc. They are accomplished with only enough planning necessary to solicit participation from the colleague and prepare the product element to a state where it can be reviewed. Such reviews should always be limited to two hours. Peer reviews usually examine a portion of a product rather than the entire document, plan, specification, or design under review. Because peer reviews impose only small blocks of time, this technique is used frequently (typically many times in a given week) among the set of people working that project. Notes are kept by the author of the product element. No "list of issues" or action items result from peer reviews.

The MDL Standards, Guidelines and Procedures (NWS 2003a) contains guidelines on how to conduct peer reviews.

6.2 CODE WALKTHROUGHS

Similar to peer reviews, code walkthroughs involve only technical staff. The number of participants, counting the product author, ranges from three to six. Scheduled code walkthroughs have a maximum duration of two hours. The focus is on identifying technical issues and concerns, not solutions. As discussed below, a moderator is assigned to keep the review focused only on technical issues, rather than discussing solutions to issues. A list of the identified issues is made during the review.

Follow-up code walkthroughs are conducted as appropriate to the importance of the identified issues. The list of issues packaged with a minimal amount of data about the review itself (e.g., date and members) are placed in the Software Development Files (SDFs) for the product that was partially or fully reviewed.

The MDL Standards, Guidelines and Procedures (NWS 2003a) contains guidelines on how to conduct code walkthroughs.

6.3 MANAGEMENT REVIEWS

Management is responsible for resolving the technical, schedule, and resource issues that are a significant risk to the project. Primary communication and resolution mechanisms used by management are Project Status Reviews.

6.3.1 PROJECT STATUS REVIEWS

Project Status Reviews occur both on a periodic and event-driven basis. Participants are the leads for the various elements of the project, for instance, Project Manager and Task Lead, should be present. Technical progress, plans, performance, and issues are discussed and tracked against the baselined plans. Each attendee presents a summary of activities and issues since the last Project Status Review as well as plans for the upcoming period. The meeting focuses on open, significant issues. Anyone can present alternative solutions for those significant issues.

Project Manager should record all issues identified at the meeting as requiring resolution. An action list is distributed by the **Project Manager** to the meeting attendees. The **Project Manager** maintains the list and detailed records of how each issue was resolved.

6.3.2 DEVELOPMENT REVIEWS

Development reviews are conducted upon the completion of a formal milestone. Development reviews include:

Requirements Review - Presentation and request for approval of requirements.

Design Review - Presentation and request for approval of design information.

Integration Readiness Review - Presentation of the readiness to proceed to an operational platform for System Integration Testing. This review should include:

- Verify that all development steps were completed
 - including code walkthroughs
- review of the code to verify it meets The MOS project standards
- review of test procedures to verify that the software was adequately tested
- perform ad-hoc testing of the software
- verify documentation is complete
 - MDL Office Note, if necessary
 - NCEP Change Request
- approve code for NCEP testing

Operational Readiness Review - Presentation of the readiness to move to the NCEP operational platform.

- review testing performed by NCEP
- verify final version of Office Note is complete, if necessary
- approve code for operational use at NCEP

7.0 TESTING

The primary goal of all of the testing activities is to identify and remove defects and to provide a standard approach for testing the MOS project software.

The MOS project Test program is based on the following key concepts:

- The testing approach is to provide an effective, repeatable, software test process which is independent of the software language, design methodology, and development environment,
- The testing scope is to identify and remove all defects and also to validate that software applications meets all requirements allocated to them,
- The testing strategy incorporates two basic points of view: functional (user's) and structural (program attributes). The testing of each application will be designed to include adequate coverage of both the functional and structural aspects,
- The testing process will essentially follow a bottom-up approach. The testing will begin at the lowest unit level and proceed upward as units are integrated into the application,
- Quality is designed into products using defined processes that are continually monitored and updated to improve their efficiency, to avoid recurring problems, and to maintain the desired quality of resulting products.

The MDL Standards, Guidelines and Procedures (NWS 2003a) describes the criteria, responsibilities and test strategy (i.e., test plans, procedures, level of testing) necessary to provide an effective, repeatable software test process which is independent of the test environment.

8.0 CONFIGURATION MANAGEMENT

The Configuration Management (CM) function ensures that the software development process is followed and that the necessary metrics are collected.

The CM program is based on the following key concepts:

- The tools used (Library Management System) are customized to the defined development life cycle,
- All code changes are documented and related to the appropriate change document, and
- The change documents are customized to collect the necessary metrics and to provide management, developers, and users with the appropriate information in a timely manner, so as to identify risks as early as possible.

The MOS Library Management System was developed to document changes that were made to codes within the operational libraries of the MOS.

No formal Configuration Management System is used to manage development software.

9.0 SOFTWARE STANDARDS

The critical importance of developing well documented and well-structured code has become more obvious with time. Except for, possibly, some small programs/subroutines written exclusively to test an idea or structure that will soon be discarded, Government developed software will be inherited and maintained by others. It is imperative to follow good coding and documentation rules in the development of all code, and in particular code that is to be handed off for use outside of MDL. Reasons include:

- With several people involved in a project, it is important that guidelines be followed so that all can easily "read" another person's program,
- Usually, it will fall to someone other than the originator to modify or maintain a program at some time in the future. Again, if a program has been written and documented according to prescribed rules, revisions and maintenance are much easier,
- Standardization will reduce errors in coding. The eye and mind become accustomed to "patterns," and a break in a pattern may be an error,
- Converting a body of software from one computer system to another is easier if it is all written and documented to the same standards, and
- New employees with little or no programming experience can be more easily trained in good procedures if those procedures are written down and everyone follows them.

In summary, the objectives of these guidelines are to enhance clarity, testability, maintainability, and person-to-person and computer-to-computer transferability of software throughout its life cycle. The following standards are contained in the MDL Standards, Guidelines and Procedures (NWS 2003a):

MDL FORTRAN Standards - The developmental software follows standards established within MDL for all FORTRAN codes. Fortran77 conventions with extensions are used on the Hewlett-Packard (HP) platform.

NCEP FORTRAN Standards - The operational software follows both MDL and NCEP standards established for FORTRAN codes. The NMC Handbook documents the NCEP standards. The internal code documentation required by the standards is enforced by an automated checking procedure before the software is implemented in operations. On the NCEP IBM platform, software must conform to Fortran95 conventions.

Scripting - UNIX scripting done for the Posix shell is used for the operational software running in the NCEP environment.

10.0 DEVELOPMENT AND TESTING ENVIRONMENT

The development environment for MDL consists of legacy HP running a variety of X software applications. The following HP equipment, broken out by floor, is used for development:

10th floor - monsoon, cirrus, blizzard, ice, precip

11th floor - chinook

The operational environment at NCEP is an IBM supercomputer. It is the IBM eServer P690 and FASTT500 Storage server that uses a FORTRAN 95 compiler on an AIX UNIX platform.

11.0 REFERENCES

National Weather Service, 2003a: MDL Standards, Guidelines and Procedures, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S. Department of Commerce, (in preparation).