

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 NDFD PROJECT**

DRAFT
February 13, 2004

TABLE OF CONTENTS

Section		Page
1.0	Introduction.....	1
1.1	Document Organization.....	1
1.2	Supporting Documents.....	2
2.0	Project Organization and Responsibilities.....	3
2.1	National Digital Forecast Database (NDFD).....	3
2.2	Other Organizations and Contractors.....	3
3.0	Project Management.....	4
3.1	Responsibilities.....	4
	Management Process.....	4
3.2.1	Planning Process.....	4
3.2	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.....	13
4.3.3	Code and Unit Testing.....	15
4.3.4	Build and Release.....	17
4.3.5	Software Integration Testing.....	19
4.3.6	System Integration Testing.....	22
5.0	Documentation.....	25
5.1	Formal Documentation.....	25
5.2	Informal Documentation.....	28
6.0	Reviews.....	29
6.1	Peer Reviews.....	29
6.2	Code Walkthroughs.....	30
6.3	Management Reviews.....	30
6.3.1	Project Status Reviews.....	30
6.4	Development Reviews.....	30
7.0	Testing.....	32
8.0	Configuration Management.....	33
9.0	Software Standards.....	34
10.0	Environment.....	35
10.1	Development.....	35
10.2	Testing.....	35
11.0	References.....	36

TABLE OF CONTENTS (continued)

Section		Page
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.....	14
Table 4.3.3-1	Code and Unit Testing Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	16
Table 4.3.4-1	Build and Release Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	18
Table 4.3.5-1	SwIT Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	21
Table 4.3.6-1	System Integration Testing Input and Output, Entrance and Exit Criteria, Process Control, and Metrics.....	24
Table 5.1-1	Documentation Chart.....	27

MDL SOFTWARE DEVELOPMENT PROCESS FOR THE NDFD PROJECT

1.0 INTRODUCTION

The Meteorological Development Laboratory (MDL) Software Development Process (SDP) establishes the software and development processes that are used throughout the National Digital Forecast Database (NDFD) project software development life cycle. This document describes the processes and procedures that are used to design, implement, and test NDFD 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 roles and responsibilities for MDL and other organizations that interface with MDL during the development process.

Section 3 - The Deployment Strategy section describes the development and release strategy for software developed by MDL under this SDP

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 AWIPS 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 AWIPS applications.

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 NDFD applications.

1.2 SUPPORTING DOCUMENTS

Other important documents related to the SDP include:

MDL Standards, Guidelines and Procedures (NWS 2002b) - This document contains the software standards used in the development of AWIPS applications 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 roles and responsibilities within MDL, and the Government and Contractor organizations that interface with MDL during the development of application software.

2.1 NATIONAL DIGITAL FORECAST DATABASE PROJECT DEFINITION OVERVIEW

The objective of the National Digital Forecast Database (NDFD) project is to implement a National Digital Forecast Database (NDFD) from which products can be produced that offer high resolution and maximum flexibility to customers and partners. The NDFD is a real-time national (CONUS) and OCONUS mosaics of gridded sensible weather forecasts generated locally at WFOs. Using the IFPS, WFOs will create forecast grids which will be transmitted on a regular basis to the NDFD central collection point. Every hour, an updated NDFD mosaic will be issued containing the most recent WFO forecast grids. When fully implemented, NDFD gridded forecast mosaics will be available to a broad spectrum of public and private users.

2.2 OTHER ORGANIZATIONS AND CONTRACTORS

MDL staff must interface regularly with other organizations that play a critical role in the development and deployment of NDFD software.

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

OST/Software Engineering Center (SEC) - SEC has overall responsibility for the AWIPS development effort, managing the distributed development across all development organizations. They have oversight over a number of technical teams such as the Software Engineering Group, Systems Engineering Team, Communications Working Group, Hardware Working Group and Performance Working Group. The SEC is responsible for reviewing the software plans for appropriateness and completeness, especially in relationship to the overall engineering plans. The SEC Systems Engineer deals with system performance issues and has authority and responsibility for making the system meet performance requirements.

OST/MDL/Product Generation Branch (PGB) - PGB is responsible for developing and implementing software that gather and transmit IFPS forecast grids to the NDFD central collection point.

OST/MDL/Evaluation Branch (EB) - EB is responsible for developing and implementing software that evaluates and verifies NDFD data.

Office of Climate, Water and Weather Services (OCWWS) - OCWWS represents the user community for the development of AWIPS applications. They determine and validate WFO user requirements, assure user requirements and feedback are incorporated into the development cycle, conduct evaluation of services and identify enhancements to be incorporated into NDFD.

Chief Information Officer (CIO) - The CIO manages the Network Control Facility (NCF) which monitors network and site performance and supports users in troubleshooting problems.

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

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. Figure 3.1-1 shows the project organization and relationship to MDL management.

Director, MDL - Overall responsibility for the software development effort.

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. The Project Manager conducts regular oversight reviews with the MDL Director.

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

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 project 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. Project planning is a continuous process. As the development progresses, certain design decisions may change the plan or schedule.

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

Create NDFD Change Request (NRC) - A NRC starts the NDFD development cycle. NRCs can be initiated by **OS or Project Manager** and are prepared for new software requirements, software enhancements, or hardware and system changes.

Create Task Development Report (TDRs) - The **Task Manager** establishes a set of TDRs based on the known RCs.

Create Software Problem Reports (SPRs) - For software deficiencies the **Task Manager** establishes a SPR for each reported software deficiency. Software deficiencies are reported as Trouble Tickets (TTs) or were created based on the testing of a previous software release.

Estimate Level of Effort (ELOE) - For each TDR and SPR the **Task 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 TDR or SPR.

Assign Target Release - The **Task Manager** assigns the TDRs and SPRs to a Target Release.

Assign Task Manager - The **Task Manager** assigns the TDRs and SPRs to a Task Manager.

Prepare a Project Tracking Information (PTI) - The **Task Manager** prepares a Staffing Plan and Development Schedule. The schedule shows development steps and checkpoints.

Approve Staffing Plan - The **Project Manager** approves the Build Plan.

Plan Development and Test Environment - The **Task Manager** works with the **CM** to determine work set to be used, how and when it will be populated, and the build environment and schedule needed. The **Task Manager** notifies the **SAM** of the test environment required and the schedule to be met.

Initiate Development - In accordance with the schedule, the **Task Manager** actions the appropriate TDRs/SPRs to begin the development cycle.

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	NDFD Change Requests (NCRs) received	OS, Project Manager
Input	NDFD Change Requests	OS, Project Manager
Output	Updated Build Plan	Task Manager
	Project Tracking Information (PTI)	Task Manager
	TDRs/SPRs assigned a target release and Task Manager	Task Manager
Exit Criteria	Build Plan is updated and approved	Project Manager
	TDRs/SPRs in Task Managers pending list.	Task Manager
	Project Tracking Information (PTI) created	Task Manager
	Approved Staffing Plan is baselined	CM
	CM environment and schedule determined	Task Manager/CM
	Test environment and schedule determined	Task Manager/SAM
Process Controls	SQA process audits	SM
	Peer reviews	Task Manager
	Metric collection, analysis and reporting	SM
Metrics	Compare estimated to actual for each step of the development process (including maintenance) - Level of Effort (LOE) - Lines of Code (LOC)	SM

3.2.2 PROJECT TRACKING AND OVERSIGHT

Software project 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 project tracking and oversight start as soon as the effort commences. The Build Plans and Development Schedules are used as the basis for tracking and oversight throughout the project. During development, actual data are 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 plans and any renegotiated commitments are revised and reviewed.

When modifying the plans, the Software Manager 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 projects.

Status Reviews - Status reviews are conducted with the Project Manager, Task Manager and developers to review progress and address issues. These can be conducted weekly or biweekly depending on the application and point in the development process.

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.

Coordination Meetings - The Hydromet Engineering Design Group (HEDGE) meeting is conducted weekly to coordinate between Project Managers, Project Leads, Task Managers, Configuration Management, and the System Administrator.

MDL Staff Meeting - The MDL staff meeting is conducted weekly to coordinate between the MDL Director, Branch Chiefs (Project Managers) and Software Manager.

In addition, following information is used by the Task Manager to provide project 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 TDRs/STDRs per release with estimated level of effort and identified Project Lead, Task Manager 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).

Audits - Verification that the SDP is being followed.

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 applications.

- Requirements Analysis
- Design
- Coding and Unit Testing
- Build and Release
- 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.

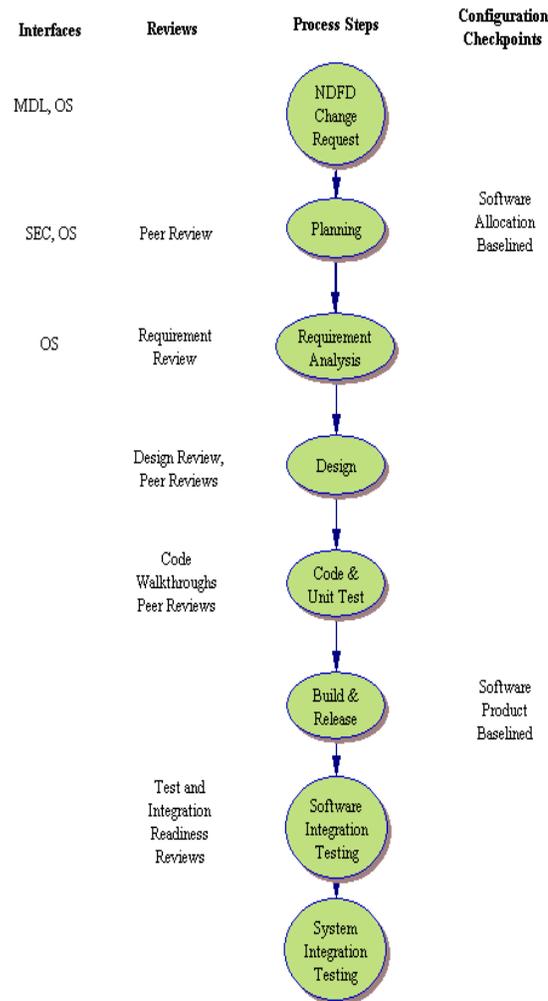


Figure 4.0-1: Data flow of the Software Development Process.

4.1 RESPONSIBILITIES

The development organizational definitions for each role along with their specific responsibilities are described below and in Figure 4.1-1. Depending on the size of the development effort, a staff member may be tasked with multiple roles.

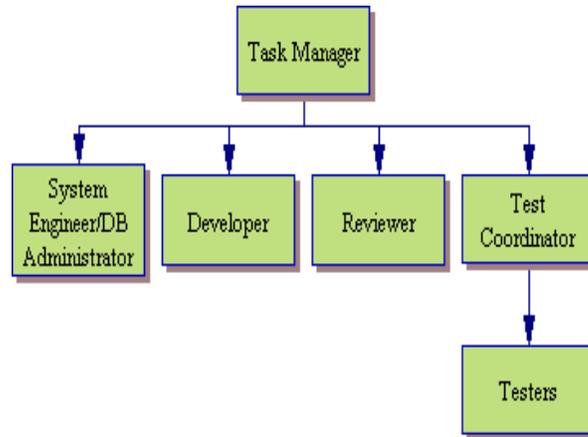


Figure 4.1-1. Development Organization

Task Manager - The Task Manager's leads the development of the TDRs/STDRs/SPRs. The Task Manager coordinate the activities of the Design Lead, Developers, and reviewers. The Task Manager performs the requirement analysis, assists with the design, reviews developers code and test procedures, and assists the Test Coordinator prepare test plans and schedules. The Task Manager reports to the Project Manager.

Software Engineer/DB Administrator - The Software Engineer is the chief architect and leads the requirement analysis peer reviews and code walkthroughs. Other responsibilities include acting as the DB Administrator. The Software Engineer reports to the Task Manager.

Developer - The Developer's major responsibility is to code software using the established coding standards, guidelines, and procedures. After completing the coding phase, the developer go through the code review process with their Task Managers. It is also the developer's responsibility to create test procedures and to perform the unit testing.

Test Coordinator - The test coordinator's responsibility includes preparing a test matrix and test schedule and ensuring that the test coverage is complete. If it is not complete, the test coordinator either will create additional test cases or will ask for the developers help to create the specific test cases. The Test Coordinator will report the test results and total test coverage to the Task Manager on an establish schedule.

Tester - A developer, assigned by the Task Manager, preferably someone other than the developer who wrote the code.

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

4.2 METRICS

The Project Manager/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 and available for the SM.

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 MDL 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 dependencies of the 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.

Also, any guidelines and process descriptions necessary to complete the activity will be referenced and kept in the MDL Standards, Guidelines and Procedures (NWS 2002b).

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 a written 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 Project Planning (Section 3.2.1) activity results in a group of TDRs being assigned to a release and a Task Manager. The Task Manager defines and develops software requirements and prepares a Requirements Document (RD).

The RD is a set of detailed software requirements derived from the Requests for Change linked to the TDRs. 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 RD can include data sources, types, and rates. A compliance matrix is used to map software and derived requirements to TDRs.

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 RD and those changes presented at the appropriate review.

A Requirement Review is held to finalize the understanding of requirements with the customer and obtain approval to develop the application or enhancement.

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 Manager** defines the set of software and derived requirements for this project.

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

Schedule Requirements Review (RR) - A RR is scheduled by the **Task Manager** to present the requirements to the user community. A Requirement Review presentation is prepared and distributed to the participant

at least one week prior to the review.

Perform Requirements Review (RR) - A RR is conducted by the **Task Manager**. Participants should include Project Manager, IFPS Project Lead, System Engineer, Software Manager, Configuration Management, DDT Team, OCWWS, and System Engineering Center (SEC). Action items (if necessary) are documented and delivered to the Project Manager and tracked through closure by the **Task Manager**.

Prepare Development Schedule - The **Task Manager** prepares an internal detailed schedule employed by the Project Manager to track development progress.

Update Project Tracking Information - The **Task Manager** updates the Project Schedule and Build Plan with any changes that result from a better understanding of the requirements. The **Task Manager** will communicate these changes to the Software Manager.

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	Build Plan is approved by Project Manager	Project Manager
Input	NCRs	OS, Project Manager, SEC
	Project tracking Information (PTI)	Task Manager
Output	Requirement Document	Task Manager
	Requirement Review documentation	Task Manager
	Updated Project Tracking Information (PTI)	Task Manager
Exit Criteria	Requirements Document is approved	OS
Process Controls	Requirements Review	Task Manager
	SQA process audits	SM
	Peer Reviews of Requirement Document	Task Manager
	Metrics, collection, analysis and reporting	SM
Metrics	To Be Determined (TBD)	SM

4.3.2 DESIGN

The Design phase will establish a complete software design to be used by the developers of the application. Use cases are created to further define the requirements. 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 application or enhancement. In most cases only one Designing step is required, however an optional step on Detail Design Process and its Review can be added for complex functionality.

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

Partition software - The **Task Manager** allocates requirements to high level software functional items. The **Task Manager** establishes a set of Sub-Task Development Reports (STDRs) related to a TDR.

Prepare design - The **Developers** defines component interfaces, relationships and data flows, physical structure, user interface, data structure, and critical test scenarios.

Prepare Software Design Document - The **Task Manager** prepares a Software Design Document.

Peer Review - Peer reviews are conducted by the **Developers** to review selected design components and documents. Participants could include the Task Manager and development team.

Schedule Design Review - A design review is scheduled by the **Task Manager** to present the design to the user representatives, system engineering and integrator community. A Design Review presentation is prepared and distributed to the participant at least one week prior to the review.

Perform Design Review - A design review is conducted by the **Task Manager**. Participants should include Project Manager, IFPS Project Lead, System Engineer, Software Manager, Configuration Management, OCWWS, and System Engineering Center (SEC), and the development team. Action items (if necessary) are documented and delivered to the Project Manager and tracked through closure by the **Task Manager**.

Assign developers - Based on the Build Plan and Staffing Plan, the **Task Manager** assigns developers to the STDRs/SPRs and actions them to "Development."

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

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 Document is approved	OS
Input	Requirement Document	Task Manager
	Requirement Review documentation	Task Manager
	TDRs/SPRs	Task Manager
Output	Design Document	Design Lead
	Design Review Document	Design Lead
	Design review action items	Task Manager
	STDRs/SPRs	Task Manager
Exit Criteria	Completion of design documentation	Design Lead
	Peer review conducted	Design Lead
	Design review conducted and approved	Task Manager, OS, SEC
	STDRs assigned to the developer(s)	Task Manager
Process Control	Design Review	Task Manager
	SQA process audits	SM
	Peer Reviews	Task Manager
	Metrics, collection, analysis and reporting	Task Manager
Metrics	TBD	SM

4.3.3 CODE AND UNIT TESTING

Once the Design Review is approved and the Design Documents have been completed, the developers assigned to the Coding activity can begin.

Coding is performed uniformly across software products using defined standards and guidelines. See the MDL Standards, Guidelines and Procedures (NWS 2002b) for the applicable 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, where possible, without any warning or error messages
- Follow good coding techniques
- Includes proper internal documentation
- Follows reasonable and understandable size limitations
- 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 2003b)

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. See the MDL Standards, Guidelines and Procedures (NWS 2002a) for more details on how to prepare test procedures.

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 the MDL Standards, Guidelines and Procedures (NWS 2002a).

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

Fix Defects - Once defects found during the Code Walkthrough are fixed by the **Developer** and approved by the **Reviewers**, the **Developer** can check the code in for Task Manager's Review.

Software Review - The **Task Manager** reviews the code and test procedures and works with the Developer to address any issues.

Arrange Build schedule with CM - A build schedule for testing is coordinated between **CM**, and the **Task Manager**.

Conduct a Test Readiness Review (TRR) - The **Task Manager** will conduct a review of the coding activities and determine the readiness for Software Integration Testing (SwIT). Participants should include Project Manager, Task Manager, Software Manager, Configuration Management, and

development team members. Action items (if necessary) are documented.

Task Manager Approval - If the software is approved at the TRR, the **Task Manager** will action the STDRs to "approved."

Notify CM - The **Task Manager** will notify CM that the software is ready for build, release and testing.

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	Approved design documents	Task Manager
	Change documents (SPR/STDRs) delegated to Developer	Task Manager
Input	Design Documents	Task Manager
	Use Cases	Task Manager
	STDRs/SPRs under work	Task Manager
Output	Checked-in code	Developer
	Completed Code Walk through form	Developer
	New or updated Test Procedures	Developer
	List of Test Procedures for integration testing	Developer
	Reviewed STDRs/SPRs in CM's pending list	Task Manager
Exit Criteria	Completion of Unit Testing	Developer
	Completion of the Code Walkthrough	Developer
	Completion of Test Procedures	Task Manager
	Approved status on STDR/SPRs	Task Manager
	Completion of Peer Reviews	Task Manager
	Delegate Tester for STDR/SPR	Task Manager
	Successful nightly development builds	Task Manager
Completion of Test Readiness Review	Task Manager	
Process Controls	Management Reviews	Task Manager
	SQA process audits	SM
	Peer Reviews of Test procedures	Developer
	Code Walkthroughs	Developer
	Metrics, collection, analysis and reporting	Task Manager
Metrics	TBD	SM

4.3.4 BUILD AND RELEASE

Once the Task Manager's review is completed, the CM is notified that the code is ready for a CM build. The Build and Release step includes building, preserving, releasing and staging the software. This section only covers preparation and delivery of the software to internal sources for verification and validation purposes.

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

Print out STDR/SPR for verification - **CM** prints out all related STDRs/SPRs which have been approved for incorporation into the CM build.

Verify each approved software "item" is related to an approved STDR/SPR - **CM** verifies all approved items are exported to the proper CM work set and related to one of the approved STDRs/SPRs. **CM** then actions all approved STDRs/SPRs and the related items to the "Build_Release" state.

Initiate CM Build - **CM** incorporates all software items at "Build_Release" into the CM build trees on HP and Linux, as appropriate, and initiates the build process. Upon completion of the build process, **CM** checks the build log for errors to verify that the build was successful. If errors are found, the Release Coordinator is notified.

Preserve all executables and shared libraries - **CM** will initiate a process to preserve all executables and shared libraries from the successful build into PVCS and will verify that there were no errors reported in the preserve log.

Create Baselines - **CM** will create baselines containing all project deliverables and source code and will confirm that the baselines were created successfully.

Create Release from successful baseline - **CM** will create a release of the software deliverables contained in the baseline. **CM** will verify that all software deliverables have been placed in their proper runtime directory structure.

Stage release output - **CM** will 'stage' the release output by packaging it into a tar file(s) in preparation for delivery and installation.

Email Task Managers when software release is ready - **CM**, once satisfied that the software has been successfully baselined, released and staged, will notify the Release Coordinator and Task Manager(s) that the software package is ready to be burned on a CD and/or installed on a test machine.

Action STDRs/SPRs and software items to next state - **CM** will action all STDRs/SPRs from the status of "Build_Release" to "Test" and all software items from the status of "Build_Release" to "Delivered."

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

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

Title	Description	Responsibility
Entrance Criteria	Updated software items at "approved" state and in appropriate work set All approved items are related to approved STDRs/SPRs	Task Manager and verified by CM
Input	Approved SPRs/STDRs	Task Manager
Output	Executables and shared libraries are preserved in PVCS. Baseline is created in PVCS Deliverables are released to the appropriate release directories Email notification sent to Release Coordinator and Task Manager(s) New tar and install files have been created in staging area STDRs/SPRs at "Test" state Updated software items at "Delivered" state	CM CM CM CM CM CM
Exit Criteria	Log files without errors from each step STDR/SPR printouts given to the Release Coordinator and Task Manager(s)	CM CM
Process Controls	SQA process audits Metrics, collection, analysis and reporting	SM Task Manager
Metrics	TBD	SM

4.3.5 SOFTWARE INTEGRATION TESTING (SWIT)

The purpose of Software Integration Testing is to detect functional errors, and demonstrate interface compatibility by testing the integrated application. This will involve testing the integration of all software modules which comprise the application. SWIT takes place in the development environment. Development Test Beds have been identified machines and have the latest release installed. During this test cycle newer builds of the same version number may be installed to include new code or correct defects.

To begin this phase of the testing process, the developer(s) must identify functional test procedures to validate the software requirements. These may be derived from the same test procedures used for unit testing, but must address the integrated aspects of the application. For each test procedure, the developer must write the test descriptions, obtain test data, and schedule a review of the test procedure. For large pieces of code involving new functionality, a Test Readiness Review should be held to verify the completeness of the test procedures before approving the software for software integrated testing. For small enhancements or bug fixes it is sufficient for the Task Manager or their appointee to review the test procedures before approving the associated change document.

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

Create CD from staging - The **Task Manager** will create the CD from the staging area to install on NHD# machines after they are informed by CM that the staging is ready.

Verify test environment and AWIPS configurations - The **System Administrators** will install the AWIPS configuration baseline for testing machines that the Project Lead indicates is necessary for testing. The **Task Manager** reviews the test environment and AWIPS configuration.

Prepare and Test Install and Uninstall scripts - The **Task Manager** will prepare, run and test the install and uninstall scripts during the installation cycle.

Create Test Plan - The **Task Manager and Test Coordinator** will create the Test Plan that describes the testing to be performed during SWIT. The Test Plan includes a test schedule and assigns test procedures and SPRs to the available developers for testing. See the MDL Standard, Guidelines, and Procedures (NWS 2003a) for guidelines concern the creation of a test plan.

Test Plan Peer Review - The **Task Manager and Test Coordinator** should conduct a peer review of the test plan.

Execute Test Plan/Run Test Procedures - The **Developers** run the assigned test procedures for the STDRs/SPRs and record results. New SPRs are created for all failed test procedures. See the MDL Standard, Guidelines, and Procedures (NWS 2003a) for guidelines concern test procedures.

Assign newly created SPRs to appropriate release. - The **Task Manager** will review, approve or reject, reassign or divert to another release all pending SPRs.

Prepare Test Log - The **Test Coordinator** prepares a test log showing the testing performed and the results of that testing. See the MDL Stan-

Standard, Guidelines, and Procedures (NWS 2003a) for guidelines concern the test log.

Prepare Draft User Documentation - The **Task Manager** prepares a first draft of the User Documentation.

Prepare Draft Release Note - The **Task Manager** prepares the first draft of the Release Notes.

Prepare Draft System Design and Operations Manual - The **Task Manager** prepares a first draft of the User Documentation.

Prepare Draft NCF Support Procedures - The **Task Manager** prepares an NCF support procedures document. This document identifies and describes system support procedures for the NDFD Central Server System. These procedures are used by NCF staff providing 24/7 support for NDFD

Conduct an Integration Readiness Review (IRR) - The **Task Manager** will conduct a review of the SwIT activities and determine the readiness for System Integration Testing. Participants should include Project Manager, Task Manager, System Engineer, Software Manager, Configuration Management, and development team. Action items (if necessary) are documented.

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

Table 4.3.5-1. SWIT Input and Output, Entrance and Exit Criteria, Process Controls, and Metrics

Title	Description	Responsibility
Entrance Criteria	STDR/SPR printout for testing	CM
	CM informs RC that staging is ready	CM
Input	New tar and install files	CM
	Test Procedures	Developer
Output	Test Plan	Test Coordinator
	Updated Test Procedures	Developers or Testers
	Completed Test Log	Test Coordinator or Task Manager
	SPRs assigned to a release	Task Manager
	Draft User Documentation	Task Manager
	Draft Release Notes	Task Manager
	Draft Installation Instructions	Task Manager
	Draft NCF Support Procedures	Task Manager
Exit Criteria	Draft Systems Design and Operations Manual	Task Manager
	CD with the latest release/version with install and uninstall scripts	Task Coordinator
	Test log is completed, all "Fail" test procedures have an SPR # assigned to it	Test Coordinator or Task Manager
	New SRS assigned to a release	Task Manager
Process Control	Conduct an Integration Readiness Review	Task Manager
	SQA process audits	SM
	Management Reviews	Project Manager
	Metrics, collection, analysis and reporting	Task Manager
Metrics	Formal review	Task Manager
	TBD	SM

4.3.6 SYSTEM INTEGRATION TESTING (SIT)

The purpose of System Integration Testing (SIT) is to independently validate the software requirements in an environment which simulates WFO operational capabilities. SIT includes functional as well as performance and backup scenarios. This level of testing is performed after the software deliverables have been baselined under CM control, successfully passed all software integrated testing, and installed in the NDFD operational systems.

If errors are found in the application during this testing process, the Tester prepares a SPR to document the error. SPR is provided to Task Manager for resolution. The Task Manager assigns the SPRs either back to the developers to correct the error for the current release or assigns a future release to the SPR for resolution. After the application has been modified to address the SPRs, the updated code is resubmitted to CM, new media is cut and loaded back onto the operational platform. This process is repeated until all the test procedures and change documents have been successfully executed.

The results of all formal application testing is provided to the Task Manager. This report contains the test procedures, test data, and test results, including a history of failed test cases. Copy of the completed Test Plan is included in the SDFs. SIT is considered complete after all of the test procedures have been successfully executed and all pending SPRs have been reassigned to another release.

SIT takes place in operational environment. Operational environment is identified as the NCF machines. This testing takes place after the code freeze has been implemented and the latest builds from the development environment have been updated.

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

Create CD from staging - The **Task Manager** will create the CD from the staging area to install on NHDA or NMTW machines after they are informed by CM that the staging is ready.

Test Install and Uninstall scripts - The **Task Manager** will run and test the install and uninstall scripts during the installation cycle.

Create Test Plan - The **Task Manager and Test Coordinator** will create the Test Plan that describes the testing to be performed during SIT. The Test Plan includes a test schedule and assigns test procedures and SPRs to the available developers for testing. See Section 6.1 of the MDL Test Process for AWIPS (NWS 2003).

Run Test Procedures - The **Developers** run the assigned test procedures for the STDRs/SPRs and record results. New SPRs are created for all failed test procedures. See Section 6.2 of the MDL Test Process for AWIPS (NWS 2003).

Assign newly created SPRs to appropriate release. - The **Task Manager** will review, approve or reject, reassign or divert to another release all pending SPRs.

Prepare Test Log - The **Test Coordinator** prepares a test log showing the testing performed and the results of that testing. See Section 6.1 of the MDL Test Process for AWIPS (NWS 2003).

Prepare Final User Documentation - The **Task Manager** prepares a first draft of the User Documentation.

Prepare Final Release Note - The **Task Manager** prepares the first draft of the Release Notes.

Prepare Final System Design and Operations Manual - The **Task Manager** prepares a first draft of the User Documentation.

Prepare Final NCF Support Procedures - The **Task Manager** prepares an NCF support procedures document. This document identifies and describes system support procedures for the NDFD Central Server System. These procedures are used by NCF staff providing 24/7 support for NDFD

Table 4.3.6-1 describes the input, output, entrance and exit criteria, process control and metrics for this SIT step along with the responsible parties.

Table 4.3.6-1. System Integration Testing Input, Output, Entrance and Exit Criteria, Process Controls and Metrics

Title	Description	Responsibility
Entrance Criteria	STDR/SPR printout for testing All STDRs/SPRs are closed New tar and install files are ready in staging area for CD creation	Task Manager or Test Coordinator Task Manager CM
Input	Master CD based on latest release Test Plan Updated Test Procedures Draft User Documentation Draft Release Notes Draft Installation Instructions Draft NCF Support Procedures Draft Systems Design and Operations Manual	CM and Release Coordinator Test Coordinator Test Coordinator Task Manager Task Manager Task Manager Task Manager Task Manager
Output	Test Log Master CD Installation Instructions Release Notes User Guide NCF Support Procedures Systems Design and operations Manual	Test Coordinator Task Manager Task Manager Task Manager Task Manager Task Manager
Exit Criteria	CD with the latest build/version with install and uninstall scripts All STDRs/SPRs are tested and results are recorded on the log Test Log is complete, all "Fail" test procedures have a SPR # assigned to it	Task Manager Developers or Testers Test Coordinator
Process Control	SQA process audits Management Reviews Metrics, collection, analysis and reporting	SM Project Manager Task Manager
Metrics	Defect Analysis Counts of Pass/Fail test procedures	Test Coordinator Test Coordinator

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. It is included as a separate section in this document to emphasize its importance.

5.1 FORMAL DOCUMENTATION

Documentation is prepared in accordance with the guidelines provided in the MDL Standards, Guidelines and Procedures (NWS 2002b). Documentation will be updated as appropriate. Application documentation is updated for each release of the software. Development documentation is updated continuously and archived in each application Software Development Files (SDFs). Management and project tracking information is kept in the Project Management Files (PMFs).

This section identifies what documents are produced during the development life cycle. See the MDL Standards, Guidelines and Procedures (NWS 200b) and the examples section of the MDL Development Library for more information concerning the following documentation.

Task Development Report (TDRs) - Formal configuration item used to track AWIPS development tasks. These tasks are commonly included in the MDL Staffing Plan and are assigned to a Project or Task Manager.

Sub-task Development Report (STDR) - Formal configuration item used to assign development tasks to individual developers.

Software Problem Report (SPRs) - Formal configuration item to track software deficiencies and are assigned to a specific release and developer.

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 TDRs/STDRs per release with estimated level of effort and identified Project Lead, Task Manager 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).

Requirements Document (RD) - Formal documentation of the requirements agreed to with OS. These requirements are trackable to all aspects of the development cycle.

Software Design Document - The purpose of the Software Design Document is to describe the software architecture, processes, databases, data flows, and structure. Documentation may include:

- requirements overview
- software system overview
- descriptive overview of Process and Elements
 - site-level diagrams, Data Flow Diagrams (DFDs).
- Detailed Design
 - Process name
 - executable name
 - file system I/O
 - Process schedule
 - Data Flow Diagrams (DFDs).
- NDFD Database Table descriptions

Test Plan - The Test Plan defines the criteria and procedures for testing the NDFD prior to declaring software operationally ready. See the Test Process for NDFD (NWS 2003c), Section 6.1 for a detailed description of the test plan.

Test Procedures - For a given unit, module, or application, 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. See the Test Process for NDFD (NWS 2003c), Section 6.2 for a detailed description of the test procedures.

Installation Instructions - Instructions used to install NDFD software. The Installation Instructions usually includes the following sections:

- Prerequisites
- Pre-install Preparations
- "The" Install Instructions
- Post-Install Instructions
- Deinstall Instructions
- Attachments
 - New/Merge/Replace File List
 - Install Space Requirements
 - Sample Install/Deinstall Logs

User Documentation - Web and hard copy documentation prepared to instruct users how to use and configure application software. The User Documentation usually contains documentation on all functionality, but for each specific release, it features and highlights what's new in the release. The new or updated functionality usually takes the form of one or more of the following:

- Overview
- Technical
- Customization
- Troubleshooting
- FAQ

System Design and Operations Manual - This manual serves dual purposes. One is to provide a description of the system architecture and design of the NDFD Central Server System (CSS). The other is to describe the normal operations of the system.

- Purpose
- System Description
- System Architecture diagrams
- System Design
 - High level system design concept
 - Detailed design and operation
 - Failover mode
- Software Overview
 - Executables
 - Data inventory
- Processing Overview
- System Level Interfaces
- Database Backup procedures
- Restoration of NDFD from Image Disks

NCF Support Procedures - This document identifies and describes system support procedures for the NCF. The tasks fall into 4 general categories: monitoring of NDFD, system diagnostic procedures, and recovery procedures and troubleshooting based on calls from the WFOs.

- Overview
- Monitoring
 - Hardware Monitoring
 - Software Monitoring
 - Troubleshooting
- Specific Diagnostic and recovery procedures

Table 5.1-1 summarizes the procedures and responsibilities for producing/updating, reviewing, and approving. It also indicates which documents are under configuration management.

Table 5.1-1. Procedures and Responsibilities for Producing/Updating, Reviewing, Approving, Controlling Documentation

Document	Preparer	Approving Power	Under CM
TDR	Task Manager	Project Manager	Yes
SPR	Task Manager	Project Manager	Yes
Project Planning Information (PTI)	Task Manager	Project Manager	Yes
Requirements Document	Task Manager	Project Manager	Yes
Design Documentation	Task Manager	Project Manager	Yes
STDR	Task Manager	Project Manager	Yes
Test Plan	Task Manager	Project Manager	Yes
Test Procedures	Developers	Task Manager	Yes
Installation Instructions	Task Manager	Project Manager	Yes
User Documentation	Task Manager	Project Manager	Yes
System Design and Operations Manual	Task Manager	Project Manager	Yes
NCF Support Procedures	Task Manager	Project Manager	Yes

5.3 INFORMAL DOCUMENTATION

A software project's informal documentation consists of a set of Software Development Files (SDFs). The SDF contains 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. As a minimum, each SDF will contain the following:

- formal documentation and a master document list,
- Briefing slides,
- Design information and documentation, including rationale supporting design decisions,
- Peer review, design review, and code walkthrough results, checklists, and comments,
- Test Procedures and Test Logs
- All engineering and formal CM change history,
- Meeting minutes, memos, action items, checklists and correspondences, and
- A log of technical lessons learned that could be of value in the future to this product or other efforts.

6.0 REVIEWS

The purpose of this section is to identify the types of reviews employed and when they occur. The guidance addressing review frequency, preparation, participant number and characteristics, formality, and recording and closeout of issues can be found in the MDL Standards, Guidelines and Procedures (NWS 2002b) document.

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
- Project Management Reviews
- Development Reviews

Some of these reviews are conducted internal to MDL and, in most cases, do not include customer participation. Typically, the reviews addressed in this section lead up to formal reviews (e.g., design reviews) and formal interaction with the customer.

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.

6.1 PEER REVIEWS

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(s) 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(s), item by item, who in turn raises questions and suggestions. Application of the concept is simple and inexpensive. Peer reviews are ad hoc. They are accomplished with only enough planning necessary to solicit participation from the colleague(s) and prepare the product element to a state where it can be reviewed. Such reviews should always be limited to two hours. To accomplish that 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 among the set of people working that project. Notes, issues and action items are kept by the author of the product element. No formal "list of issues" or action items result from one-on-one reviews. 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). Management personnel track these reviews and ensure they are occurring through project status reviews.

The MDL Standards, Guidelines, and Procedures (NWS 2003b) contains guidelines for conducting Peer Reviews.

6.2 CODE WALKTHROUGHS

Similar to peer reviews and code walkthroughs involve only technical staff. The number of participants, counting the product author, ranges from three to six. Scheduled peer reviews 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.

Follow-up meetings are conducted as appropriate to the importance of the identified issues. All code walkthrough documentation are placed in the Software Development Files (SDFs) for the product that was partially or fully reviewed.

The MDL Standards, Guidelines and Procedures (NWS 2002b) contains a set of sample checklists that can be used to ensure a consistent and relatively complete review of various software products, and a sample form that can be used to record and track comments and issues generated during scheduled peer reviews.

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 and line management reviews, which includes Program Management 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, Task Manager, DDT team members, System Administration (SA), Software Manager (SM) and Configuration Management (CM) always should be present. Technical progress, plans, performance, and issues are discussed and tracked against the baselined project 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.

6.4 DEVELOPMENT REVIEWS

Development reviews are conducted upon the completion of a formal milestone and should be performed in accordance with a documented procedure contained in the MDL Standards, Guidelines and Procedures (NWS 2002a). A key purpose of the review is to present the accomplishments and results of the software project up to the current milestone with other NWS organizations. In many cases approval may be required to proceed to the next step. Review material focuses on the key points of the project activities and project status. Activities performed since the last review, planned activities, and the status of all open items identified in previous reviews are discussed. In addition a standard element of all preparatory reviews is explicit attention to project risks including schedule, resources, and technical.

Development reviews include:

Requirements Review - Presentation and request for approval of requirements.

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

Test Readiness Review (TRR) - The purpose of the Test Readiness Review (TRR) is to evaluate the software, development process, and test procedure developed during the informal application testing to certify that all software requirements are properly validated. The TRR is performed by the Task Manager(s), the developers, and Software Manager and indicates readiness for Software Integration Testing (SwIT). The evaluation process involves reviewing the test procedures and available test data for each test procedure. Requirement and Design documents along with code walkthrough results are used as supporting information during the evaluation process. Upon approval, the test procedures are submitted for formal application testing.

The TRR includes:

- Verify that all development steps were completed
- Verify documentation is complete
 - Installation Instructions
 - Test Procedures
- Verify building and release information
- Coordinate build schedule with CM
- Verify test environment and AWIPS configurations are ready on the test bed

Integration Readiness Review (IRR) - The purpose of the Integration Readiness Review (IRR) is to evaluate the application's readiness to proceed to an integration platform for System Integration Testing. The IRR is performed by the Task Manager(s), the developers, and Software Manager. The evaluation process involves reviewing the test results of Software Integration Testing and to ensure all defects are either corrected and tested or have been reassigned to a later release. A satisfactory resolutions of all change documents along with any open SPRs is essential before approval.

The IRR includes:

- Verify software was adequately testing
 - Review of the Test Plan
 - Review of test logs
 - Review and document outstanding problems
- Verify documentation is complete
 - User Documentation
 - System Design and Operations Manual
 - NCF Support Procedures

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 MDL software applications.

The MDL 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 meet 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,
- The applications will be handed over to NGIT for the final integration into the AWIPS system, and
- 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.

These concepts are accomplished by implementing the following:

- Informal testing,
- Test Readiness Review,
- Formal testing,
- Configuration management,
- Non-conformance reporting and corrective action, and
- Training.

The MDL Test Process which is part of the MDL Standards, Guidelines, and Procedures (NWS 2003a) describes the criteria and test strategy (i.e., test cases, 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 MDL Configuration Management (CM) function ensures that the software development process is followed and that the necessary metrics are collected.

The MDL CM program is based on the following key concepts:

- The tools used (PVCS Dimensions for AWIPS) 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.

These concepts are accomplished by implementing the following:

- Creating the necessary development and build environments,
- Assigning the correct roles for the development/test staff,
- Updating the change documents to support the project,
- Establishing the daily development builds,
- Creating the test releases, and
- Creating the baselines for alpha test and full deployment.

The Configuration Management Plan for AWIPS (NWS 2003b) provides the detailed configuration management procedures applicable to deliverables: code, associated data files and documentation, responsibilities, tools and techniques used, the stage at which items are brought under configuration control and change control management.

10.0 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,
- Code developed by the development team is for the express purpose of implementation and integration into a much larger system. If all such code follows the same guidelines, understanding and dealing with it will be much easier, and we will be able to answer questions more readily than otherwise,
- 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):

- C,
- C++,
- ESQL,
- scripts, and
- HTML.

10.0 ENVIRONMENTS

10.1 DEVELOPMENT

OST software development, software compilation and builds are performed on the following two suites of Software Development and Build systems. These systems are NOT on the AWIPS WAN. They have address space on the SSMC2 Building LAN, and they are protected by OST managed firewalls. The NHD (National Headquarters Development) Suite of Software Development and Build support MDL software development. This suite of systems is located in the 7th floor computer area in SSMC2 and in the MDL computer areas on the 10th floor of SSMC2.

The MDL environment is divided into the developers, workspace and the project workspace. The coding, development, and testing of units and modules will be conducted in the developers' workspace under control of the developers. The testing of the integrated modules which comprise the application will be conducted in the project workspace.

10.2 TESTING

TBD

11.0 REFERENCES

CSC, 2000: National Weather Service Advanced Weather Interactive Processing System Software Development Plan. Contract Number 263-96-D-0322, Task Number DOC-NOAA-1998-C-1523, Doc. ID No. AWIPS SDP-01-00, CSC, Rockville, Maryland.

National Weather Service, 1995a: Software Development Plan for Producing WFO Hydrometeorological Applications for AWIPS, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S. Department of Commerce.

National Weather Service, 1995b: Test Plan for Producing WFO Hydrometeorological Applications for AWIPS, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S. Department of Commerce.

National Weather Service, 2002a: MDL Configuration Management Plan for NDFD, Meteorological Development Laboratory, Office of Science and Technology, NWS, National Oceanic and Atmospheric Administration, U.S Department of Commerce, (in preparation).

National Weather Service, 2002b: 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).