Department of Commerce ● National Oceanic & Atmospheric Administration ● National Weather Service

NATIONAL WEATHER SERVICE INSTRUCTION 80-304

November 11, 2019

Office of Planning and Programming for Service Delivery

Systems Engineering

SOFTWARE DEVELOPMENT

NOTICE: This publication is available at: http://www.nws.noaa.gov/directives/.

OPR: W/OPPSD/ES (Benjie Spencer) **Certified by:** W/OPPSD (Kevin Cooley)

Type of Issuance: Routine

SUMMARY OF REVISIONS: This directive supersedes NWS Instruction 80-304, dated April 21, 2009. Changes were made to (1) Updated this directive to align to the NWS Systems Engineering approach described in 80-301(1) update Section 2 to present a refined definition of Systems Engineering relevant to the current NWS environment; (2) add new Section 3 to details for the Needs and Definition stage for software development; and (3) add new Section 4 to provide details for Software Design and Development Stage. Finally, changes were made to reflect the NWS Headquarters reorganization effective April 1, 2015.

Signed 10/28/2019

Kevin C. Cooley Date

Director, Office of Planning and Programming for Service Delivery

Contents

Table of Figures

1 Introduction

This instruction provides guidance on National Weather Service (NWS) Systems Engineering (SE) for software development. Software development is the process of developing software through successive phases in an orderly way. This process includes not only the actual writing of code but also the preparation of requirements and objectives, the design of what is to be coded, and confirmation that what is developed has met objectives. A key to good software development is to use a software development life cycle (SDLC) method, which defines the tasks performed at each step in the software development process. SDLC consists of a detailed plan describing how to develop, maintain, and replace specific software. The life cycle defines a methodology for improving the quality of software and the overall development process. Additional information about the models used within the NWS are in the system development section of this document. The development of software systems involves a series of development activities that are driven by the overarching SE approach found in NWS Instruction 80-301: *Systems Engineering Process and Life Cycle* that does not change for software.

The intent of this instruction is to provide a high-level software approach and a set of activities that provide a consistent method for software development projects and to provide practical guidance for implementing that approach. This document is a guide for system program managers on the expectations of the software development team. The project should include a software team that is well versed in all aspects of software engineering and not reliant on this document as guidance for specific software development activities. This document expands on the material presented in NWS Instruction 80-301: *Systems Engineering Process and Life Cycle*, which identifies the five stages of the NWS SE approach and 80-303 *Systems Engineering for New System Development*, which describe the activities for the first two life cycle stages: the Need and Definition Stage, and the Design and Development Stage. This instruction further describes the inputs, activities, outputs, deliverables, and reviews associated with each stage of developing software systems with emphasis placed on the Design and Development Stage. However, the instruction is not meant to be prescriptive and rigid, instead, project teams can further tailor the information in this instruction to meet the project specific needs, requirements, constraints, environment, schedule, budget and situation. For example, some software development projects require completed artifacts to be provided under one deliverable, while others require an iterative approach whereby successively larger and more complete versions of the software are built with each iteration of the process model.

Another example of flexible NWS software development is the use of open source software. Open Source Software is software for which the human-readable source code is available for use, study, reuse, modification, enhancement, and redistribution by the users of that software. In other words, OSS is software for which the source code is "open."[1] Open Source Development is a collaborative process, similar to community modeling that is a critical framework for advancement within scientific organizations. The term “Open” implies the establishment of

[1] “Department of Defense”, Clarifying Guidance Regarding Open Source Software (OSS), October 2016

legal and technical governance, sharing infrastructure, and a development process that facilitates trusted modifications and evaluations within a decentralized software development structure.

2 Overview of Engineering Life Cycle Approach

This instruction focuses on the software development portion of the NWS System Engineering lifecycle. Software development is integrated within the NWSI 80-301: *Systems Engineering Process and Life Cycle* because most, if not all, developmental systems include some amount of software, and the NWS SE life cycle approach described therein establishes requirements for *all* components of the system. Specifically, once the needs and definition stage is complete and system requirements and a primary design have been developed, an understanding of the role the software plays in the overall system has been established. This information is provided to the software development team that then focuses specifically on the software and its requirements.

Although software development is a part of the NWS SE life cycle, it requires a separate instruction because of the variety of process models that can be used for software development based on the nature of the software being developed. These various software development needs facilitate the necessity for a separate instruction for the software aspect of the development of NWS systems.

The NWS System Engineering Life Cycle Model is shown in Figure 1. The Needs and Definition Stage is typically the same for any development effort, but software-specific projects typically start to diverge somewhat from the general SE course during the design and development stage. During this stage, when requirements are being allocated to software subsystems, the software development team will use the allocated requirements and preliminary system design to create a detailed software design. The detailed software design will be used for software development and the developed code will flow back to the system level for system integration to be integrated with the rest of the system before deploying the operational system.

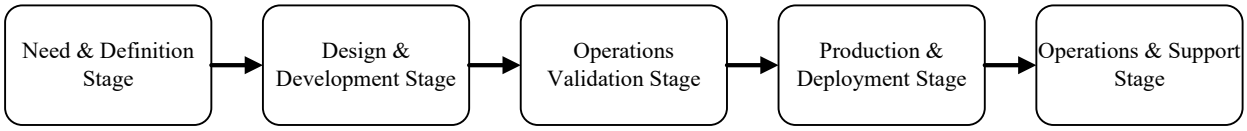

Figure 1: NWS Life Cycle Model

3 Need and Definition Stage

The SE process starts with the Need and Definition Stage that is initiated upon recognition of a need for a new or modified System of Interest. Figure 2 below illustrates the inputs, activities and outputs of this stage.

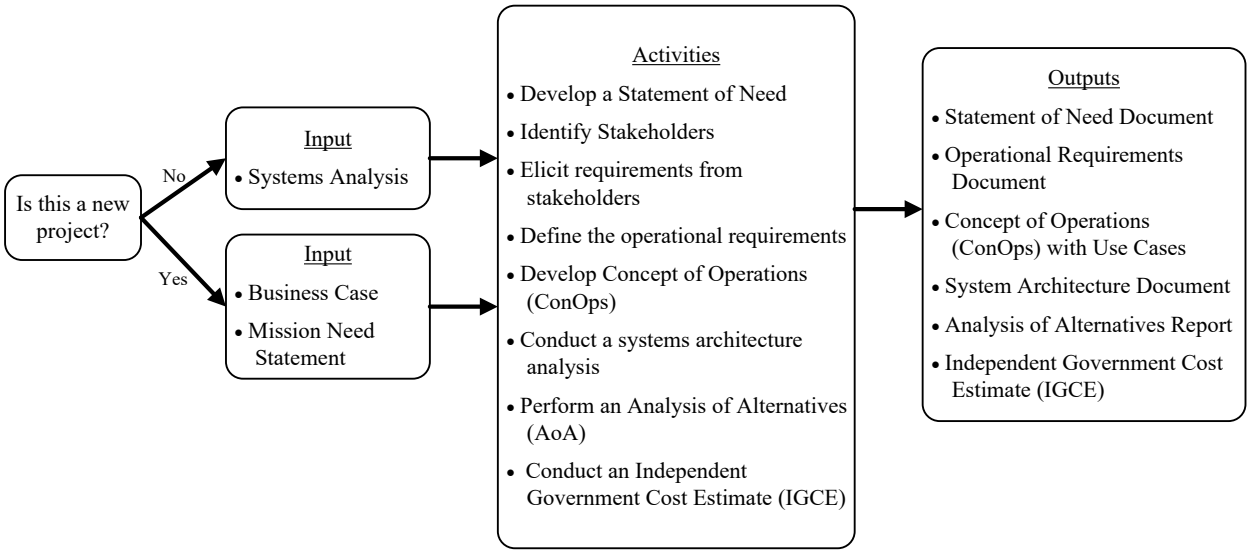

Figure 2: Need and Definition Stage

In the beginning of the Need and Definition Stage, the team members determine if the need is for the development of new software or the development of upgrades to existing software. Once the type of change is determined the team will develop a Statement of Need (SON) to document the reasons for the new/upgraded software. After the SON is complete, stakeholders have been identified and requirements elicited to support the development of the software's operational requirements. These requirements will also be used to develop the Concept of Operations (ConOps) and perform the Analysis of Alternatives (AoA) for the software system. These activities define the software needs and the output is used as the basis for designing and developing the software. More details about the activities and outputs in the Needs and Definition stage can be found in NWS Instruction 80-303 *Systems Engineering for New System Development.*

4 Design and Development Stage

The Design and Development Stage defines, validates, and realizes a System of Interest (SOI) that meets stakeholder requirements and can be produced, deployed, operated, and supported. This stage uses the outputs of the Need and Definition Stage as inputs and the primary output of this stage is the System of Interest approved for entering Operational Test & Evaluation. Figure 3 below illustrates the inputs, activities and outputs of this stage.

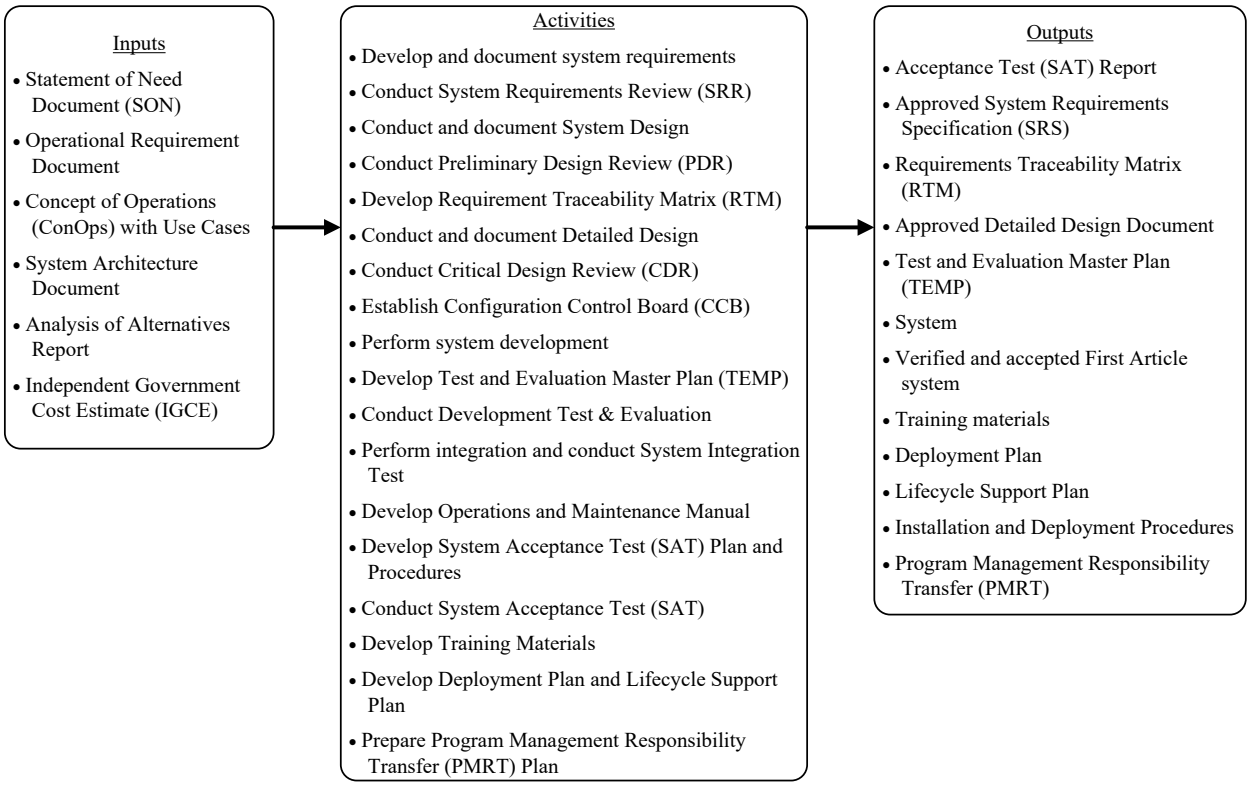

Figure 3: Design and Development Stage

The first activity of the Design and Development Stage is to Develop and Document System Requirements. During this activity, the stakeholder needs and operational requirements are used to develop functional and technical system requirements that define what the capabilities of the System of Interest are and how well the capabilities have to perform. After the requirements are developed and documented, the team conducts a System Requirements Review (SRR). The SRR is a technical review of the system requirements that examines the requirements defined for the system and ensures that an accurate, complete, and achievable set of requirements has been developed and that the selected concept will satisfy the need. SRR completion leads to conducting and documenting the system design to produce the high-level design defining the overall framework of the system and describe how its structure satisfies the requirements. The next review after the system design is the Preliminary Design Review (PDR) to demonstrate that an acceptable preliminary design is in place that will account for system requirements, thereby fulfilling the operational requirements.

Once the PDR is complete, a Requirements Traceability Matrix (RTM) is developed to document the life cycle of each requirement and provide bi-directional traceability between associated requirements. The PDR also leads to conducting and documenting the detailed design, which encompasses a description of the architecture, subsystems, components, interfaces, and data structure definitions before development begins. After the detailed design is complete, the design team conducts a Critical Design Review (CDR) to demonstrate that all system and operational requirements are within the provided risk, cost, and schedule constraints.

The CDR baselines the design and creates a need to establish a Configuration Control Board (CCB) to govern changes to and versioning of approved system changes. After CDR the team can perform system development and proceed with the full development of the system. This includes all subsystem development of hardware components, software units and/or modules, and required interface capabilities.

As the system is being developed, test planning will start. The first aspect of test planning is to develop the Test and Evaluation Master Plan (TEMP) which establishes the overall testing philosophy and strategy to be followed through operational testing, provides an integrated test program schedule, a description of the overall test and management process, and provides guidance regarding documentation and issue reporting requirements. After test planning, the team conducts Development Test & Evaluation (DT&E) to verify system development against approved system requirements. The team also engages in the activities to integrate the various system components and modules as a whole system within a controlled environment and conduct system integration test.

During system development, the team also develops the Operations and Maintenance (O&M) Manual. The O&M Manual includes detailed system installation and configuration procedures that are used in both the initial installation to support Operational Test and Evaluation (OT&E) and all subsequent field installations during full system deployment.

Once the system has been integrated, System Acceptance Test (SAT) plan and procedures are developed to document the test management, preconditions and test setup, mechanics of test performance, test schedule and required personnel, success criteria, and documentation requirements for the SAT. The SAT plan guides the team through conduct of SAT to verify that the design solution meets all of the system technical requirements and is prepared for a successful Operational Test and Evaluation (OT&E).

A successful SAT enables the team to develop training materials to ensure that end-users can be trained on how to use the system and support personnel can be trained on how to support it operationally. It also enables the development of the Deployment Plan and Life Cycle Support Plan, which describes the strategy and logistics of how, when, and by whom the deployment of the system will be executed and the strategy and logistics planning for how the system will be supported and maintained throughout its operational life cycle. Finally, during this stage, a Program Management Responsibility Transfer (PMRT) Plan is prepared to define the transfer of the system from development to an operations and maintenance organization.

More details about the activities and outputs of the Design and Development stage can be found in NWS Instruction 80-303 *Systems Engineering for New System Development.*

The following sections detail the NWS software development process within the NWS systems engineering life cycle for new systems. Although there are other activities that the software team performs during the system lifecycle, most of them are the same or very similar to the system engineering activities. These software activities are performed once the system is decomposed to software subsystems and system requirements are allocated to specific software subsystems. The following section focuses on the software activities that are vastly different from the NWS system engineering activities. In addition to the NWS software development process, Appendix

B contains a summary of the digital playbook, released by OMB, which details 13 best practices designed to help agencies deliver digital products and services quicker and more efficiently.

4.1 Perform Software Development

This activity develops the software for the system that meets the requirements and the detailed design documentation. Specifically, the software development team analyzes the technical design specifications and then develops and implements the required software capabilities. This involves coding, verification, unit testing and debugging by the developers. These activities start with creating planning documents such as a Software Development Plan (SDP). The SDP describes the developer's plan for conducting a software development effort and a sample outline of the plan is provided in appendix A. This activity also includes choosing a software development model to provide structure for how the various tasks related to software development are organized. The software development model should be selected based on the nature of the project and application. The two standard models used within NWS are the following:

- Iterative Model – This is a process where the development phases are repeated in cycles with a feedback loop after each cycle is completed. The project team learns from the preceding cycles and plans the next cycle to converge on the final implementation solution.
  - Strengths
    - Provides maximum business value within the given time and cost constraints
    - Delivers finished, tested, usable code with each iteration
    - Provides flexibility and quick feedback which is used to plan the next iteration
  - Weaknesses
    - High reliance on meaningful customer involvement leads to higher levels of risk
    - Changes in scope results in rework through multiple iterations
- Traditional Model – This is a sequential software development process, where progress flows in sequence toward the conclusion. Each phase of the development is completed before proceeding to the next phase in the sequence.
  - Strengths of using the traditional model:
    - Provides a structured, disciplined method and can be useful for maintenance projects and small projects with clearly defined and understood requirements.
    - Simple and easy to understand and use.
    - Since the phases are rigid and precise, one phase is done one at a time, it is easy to maintain.
    - The entry and exit criteria are well defined, so it easy and systematic to proceed with quality.
  - Weaknesses of using the traditional model:
    - Can prove to be a risky and inflexible model. With only a single pass through the process, integration problems often surface too late in development, and a completed product is not available until the very end of the process

- Can discourage customer involvement and lead to a system which does not meet changing customer requirements.
- Cannot adopt to the changes in requirements
- Delivery of the final product is late as there is no prototype which is demonstrated intermediately.

The traditional model is not recommended for most NWS software development projects.

4.1.1 Iterative Model

The Iterative development model is a way of breaking down the software development of a large application into smaller parts. In Iterative development, code is designed, developed, and tested in repeated cycles that each incorporate additional features until there is a fully functional software application ready to be deployed to customers. The following sections provide a high-level view of the Iterative activities but do not provide a detailed explanation of the paradigms or provide details regarding process.

4.1.1.1 Initiation

The initiation phase explores ideas and identifies a potential implementation strategy for implementing them. This phase also obtains agreement on software functions, features, and flow. Initiation will include forming the initial development team, developing a common vision for the software that will be developed, and ensuring this vision aligns with the system vision that was established in the Need and Definition Stage of the NWS System Engineering Life Cycle Model. During initiation, a release plan is developed that details when various functionality of the software will be released. As code is developed, the iterative method allows the release plan to be updated as more information is known. Initiation also defines the requirement modeling and planning with stakeholders, architecture modeling, and set up of the development environment and development tools.

4.1.1.2 Analysis

Analysis is needed to understand the ‘what’ and ‘why’ of the software that needs to be developed. The analysis also estimates the cost and prioritizes the order of the functionality and requirements. This is also where analysis modeling can be used, which does not require formal documentation, but can be documented in a variety of ways such as activity diagrams, class diagrams, and/or use case diagrams. During the analysis, the team defines the software architecture and the software usage patterns, develops a functional architecture, defines the software navigation paths, and creates a user interface mockup.

4.1.1.3 Design

During the design phase, developers and technical architects start the high-level design of the software to be able to deliver each requirement. Like the rest of the iterative method, this is an iterative process and is performed for each release. For the release, the selected architectural design defines all the components that need to be developed. In addition, the design defines user flows and database communications, as well as front-end representations and behavior of each

component. This phase will also ensure that all of the critical tools, processes, standards, and guidelines identified in the initiation phase have been put in place for the implementation phase.

4.1.1.4 Implementation

The focus of the implementation phase is to develop the software to the point where it is ready for pre-production testing. This development is based on the requirements and architecture that have been created in the design phase. The development team codes individual software units using the programming language(s) selected for the project. Specifically, the developers write source code in the selected programming language(s) that performs the logic documented during the critical design. The software code is developed and tested before being released. The releases can be internal or external releases depending on the release plans. Coding is performed in these iterative releases until all functionality of the software is developed to meet requirements.

4.1.1.5 Acceptance

The acceptance phase consists of extensive testing of the developed software, including beta testing. The results of this testing will cause fine-tuning of the software as well as rework to address defects. Once this level of acceptance testing is accomplished, the output of this phase is the complete software ready for delivery into production or integration into the larger developmental system.

4.1.2 Traditional Method

Traditional model is a software development model in which activities are divided into sequential phases with each phase consisting of series of tasks with different objectives. The phases in the traditional method produce output that becomes the input of the next phase. In addition, the traditional model allows one development phase to start only when the previous phase is complete. This sequential nature ensures each phase of the model is precise and well defined.

4.1.2.1 Initiation

The initiation phase of the traditional method builds on the information developed during the Need and Definition Stage of the NWS System Engineering Life Cycle Model. This phase leverages the artifacts of the Need and Definition Stage to plan a software solution to meet business and operational needs. The operational and system requirements are decomposed into software requirements for the software that will be developed. The initiation phase also documents the development process, best practices, and conventions that that will be used as the software is developed. The team also selects the industry standard methods for software development that will be used to document requirements, identify delivery stages, create configuration control procedures, develop technical tracking and control processes, and specify the review process. These activities form the foundation that is used for the software analysis.

4.1.2.2 Analysis

During the analysis phase, the software architecture is created. The software architecture defines an overall structure of the software and the ways in which the structure provides conceptual integrity for a system. The architecture is structured to meet all the technical and operational requirements, while optimizing the common quality attributes like performance and security. The architecture also describes the structure and organization of subsystem, the manner in which these subsystems interact, and the structure of data that is used by the subsystem. In addition to the software, the architecture accounts for any hardware, databases, and third party frameworks the software interacts with or uses. The architecture also provides a design plan that describes the software elements of a system and how they will fit and work together to fulfill the requirements of the system.

4.1.2.3 Design

The traditional design decomposes the software architecture developed in the analysis phase into software modules. Module specifications are then produced to define what each module is to do, but not how the module is to be coded. The module specifications provide a level of detail that gives objective oriented guidance to the software developers without constraining them to one particular programming algorithm. Each module created in this process will have a concise description of purpose, explanation of hierarchical relations with other modules, and a reference back to a particular requirement or set of requirements. Each module and data structure is considered individually during design with emphasis placed on the description of internal and procedural details. The team communicates the design via process flow diagrams and/or other logical, organizational documents that are developed using software modeling.

4.1.2.4 Implementation

Once the design is complete, the development team can begin coding software for the system. The development team codes individual software units using the programming language(s) selected for the project. Specifically, the developers write source code in the selected programming language(s) that performs the logic documented during the critical design. After coding is complete, developers perform unit testing to thoroughly test and identify as many defects as possible. Identified defects are analyzed and corrected, and testing is repeated until all known defects are fixed.

4.1.2.5 Software Versioning

After testing is performed and defects are resolved, the software is baselined as an approved "build" of the product. This build will be provided for integration with other hardware and software of the system. Each of these builds will go through software versioning to assign either unique version names or unique version numbers to unique states of computer software. Within a given version number category (major, minor), these numbers are assigned in increasing order and correspond to new developments in the software.

4.2 Software Configuration Management

Software Configuration Management (SCM) applies to all software development processes. SCM manages the evolution of software products, both during the initial stages of development and during all stages of maintenance. A software product encompasses the complete set of computer programs, procedures, and associated documentation and data designated for delivery to a user. All supporting software used in development, even though not part of the software product, should also be controlled with the SCM process. The SCM process evaluates, coordinates, approves or disapprove and implements changes in artifacts that are used to construct and maintain software systems. Some of the SCM activities are the following:

- **Management of the SCM Process**
  SCM controls the evolution and integrity of a product by identifying its elements; managing and controlling change; and verifying, recording, and reporting on configuration information.
- **Software configuration identification**
  Identifies items to be controlled, establishes identification schemes for the items and their versions, and establishes the tools and techniques to be used in acquiring and managing controlled items. These activities provide the basis for the other SCM activities.
- **Software Configuration Control**
  Manages changes during the software life cycle. It covers the process for determining what changes to make, the authority for approving certain changes, support for the implementation of those changes, and the concept of formal deviations from project requirements as well as waivers of them. Information derived from these activities is useful in measuring change traffic and breakage as well as aspects of rework.
- **Software Configuration Auditing**
  An independent examination of a work product or set of work products to assess compliance with specifications, standards, contractual agreements, or other criteria.
- **Software Version Control**
  Tracking of code or document modifications by each contributor for accountability and conflict resolution.
- **Software Release Management and Delivery**
  Distribution of a software configuration item outside the development activity; this includes internal releases as well as distribution to operations.

These activities are documented in a Configuration Management (CM) plan that informs everyone in the organization just how CM is carried out.

5 Operations Validation Stage

The Operations Validation Stage is the third stage in the Systems Engineering Life Cycle Model and uses the outputs of the previous stage (Design and Development Stage) as its inputs. This is the stage in which the Government formally validates software system installation and operations in the target operational environment. This includes validation that all requirements are fulfilled in order for the Government to operate and maintain the system at all sites to which it will be deployed. This is done by conducting an Operational Test and Evaluation (OT&E) of the System

and is the primary focus of the Operations Validation Stage in the SE life cycle depicted in Figure 4 below.

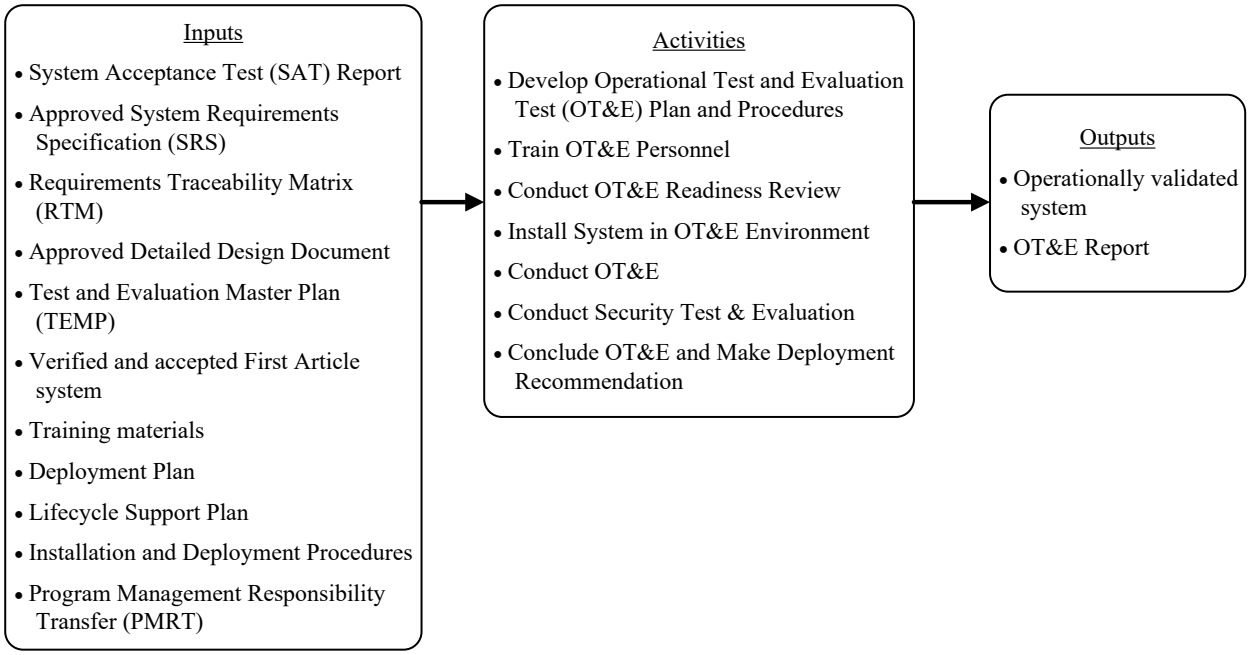

Figure 4: Operational Validation Stage

OT&E is a formal evaluation of a software system that is conducted in an operational environment and is performed after the successful completion of the System Acceptance Test (SAT) or System Test (ST) and subsequent deployment of the software system at the designated OT&E operational site(s). The subject matter presented in this section is covered in more detail in NWS Instruction 80-307 *Operational Test and Evaluation Process.*

6 Production and Deployment Stage

The Production and Deployment Stage is where the software is produced in quantity, deployed, and installed at all required operational field sites (beyond those sites involved in OT&E). The Production and Deployment Stage is depicted below in Figure 5.

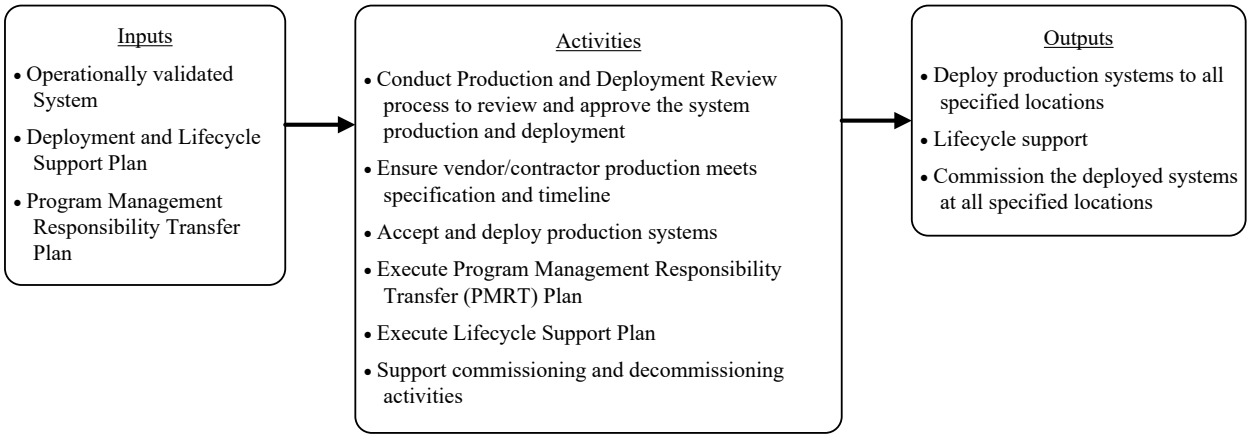

Figure 5: Production and Deployment Stage

The activities for this stage are described in greater detail in NWS Instruction 80-301 *Systems Engineering Process and Life Cycle*.

7 Operations and Support Stage

The Operations and Support Stage is where the software system is operated in its intended environment to deliver its intended services, representing the "steady state" period that lasts until the system is retired or replaced. Figure 6 below illustrates the inputs, activities and outputs of this stage. The activities for this stage are described in greater detail in NWS Instruction 80-301 *Systems Engineering Process and Life Cycle.*

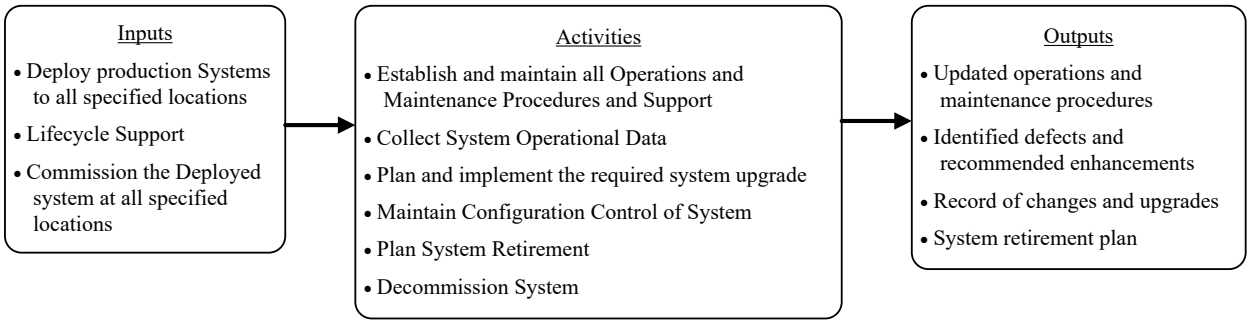

Figure 6: Operations and Support Stage

8 Roles and Responsibilities

This directive establishes the following authorities and responsibilities:

8.1 NWS Headquarters (NWSHQ)

8.1.1 Office of Planning and Programming for Service Delivery (OPPSD)

The OPPSD will collaborate with other Portfolio Offices and the Office of the Chief Information Officer (OCIO) to establish NWS policies and procedures for systems engineering. The OPPSD will provide oversight and assessment on how the established NWS systems engineering policies and procedures are applied to the NWS projects and systems. As requested and agreed, the OPPSD will also provide the systems engineering support to the projects or systems at stage(s) of life cycle process.

8.2 NWS Regional Headquarters (RQH) and NWS Regional Offices

Each Region and Office will use established systems engineering policies and procedures to their projects and systems.

8.3 Project Management Offices (PMOs)

Each PMO is responsible for ensuring the systems engineering be conducted in accordance with the established policies and procedures. If required, the PMO can request the exceptions of certain NWS SE life cycle activities through NWS Chief Engineer and/or the OPPSD director for approval.

8.4 System Owners

Each System Owner is responsible for ensuring the systems engineering be conducted through entire life cycle in accordance with established policies and procedures.

9 References and Glossary

This policy directive is supported by the references and glossary of terms listed in Attachment 1.

Attachment 1

REFERENCES AND GLOSSARY OF TERMS

References

National Weather Service Policy Directive 1-10, Managing the Provision of Environmental Information

NAO 212-15: Management of Environmental Data and Information

NWS Policy Directive 10-1, *NWS Requirements, Operations and Services Improvements*

NWS Instruction 10-103, *Operations and Services Improvement Process Implementation*

NWS Policy Directive 80-1, *Acquisition Program Management*

NWS Instruction 80-3, *Systems Engineering*

NWS Instruction 80-301, *Systems Engineering Process and Life Cycle*

NWS Instruction 80-303, *Systems Engineering for New System Development*

NWS Instruction 80-305, *Test and Evaluation*

NWS Instruction 80-306, *System Acceptance Test (SAT) Process*

NWS Instruction 80-307, *Operational Test and Evaluation (OT&E) Process*

NWS Policy Directive 80-4, *Science and Technology Planning and Programming*

NWS Policy Directive 80-5, *Science Review and Approval*

NWS Policy Directive 60-7, *Information Technology Security Policy*

NOAA Administrative Order 212-13, *Information Technology Security Management*

NIST Special Publication 800-37, *Guide for the Security Certification and Accreditation of Federal Information Systems*

NIST 800-53, *Recommended Security Controls for Federal Information Systems*

NIST 800-64, Revision 2, *Security Considerations in the System Development Life Cycle*

NIST 800-30, *Risk Management Guide for Information Technology Systems*

Glossary

The following is a list of common terms and acronyms used within the Systems Engineering industry. While many of these terms are not mentioned within the body of this document, they are nonetheless important to understanding this instruction.

Acceptance criteria - The criteria that a software component, product, or system must satisfy in order to be accepted by the system owner or other authorized acceptance authority.

Acquisition - The acquiring by contract with appropriated funds of supplies or services (including construction) by and for the use of the Government through purchase or lease, whether the supplies or services are already in existence or must be created, developed, demonstrated, and evaluated

Analysis - Use of mathematical modeling and analytical techniques to predict the compliance of a design to its requirements based on calculated data or data derived from lower system structure end product validations.

Analysis of Alternatives (AoA) - A formal analysis method that compares alternative approaches by estimating their ability to satisfy mission requirements through an effectiveness analysis and by estimating their life-cycle costs through a cost analysis. The results of these two analyses are used together to produce a cost effectiveness comparison that allows decision makers to assess the relative value or potential programmatic returns of the alternatives.

Application - Software or systems products designed to fulfill specific needs.

Assumption - A condition that is taken to be true without proof or demonstration.

Baseline - A set of configuration items (hardware, software, documents) that has been formally reviewed and agreed upon, that serves as the basis for further development, and that can be changed only through formal change control procedures.

Code - Computer instructions and data definitions expressed in a development language or in a form that is output by an assembler, compiler, or other translator.

Code Review -A meeting at which software code is presented to project personnel, managers, users, or other functional areas for review, comment, or approval.

Component - One of the parts that make up a system. A component may be hardware, software, or firmware and may be subdivided into other components.

Concept of Operations (ConOps) -The ConOps describes how the system will be operated during the life-cycle phases to meet stakeholder expectations. It describes the system characteristics from an operational perspective and helps facilitate an understanding of the system goals. It stimulates the development of the requirements and architecture related to the user elements of the system. It serves as the basis for subsequent definition documents and provides the foundation for the long-range operational planning activities

Configuration Control Board (CCB) - A group of people responsible for evaluating and approving/disapproving proposed changes to configuration items, and for ensuring implementation of approved changes.

Configuration item -An aggregate of hardware, software, or documentation components that are designated for configuration management and treated as a single entity in the configuration management process.

Configuration Management - A discipline that effectively controls and manages all modifications to system components, product, or system.

Constraint - A restriction, limit, or regulation that limits a given course of action or inaction.

Critical Design Review (CDR) - A review that demonstrates that the maturity of the design is appropriate to support proceeding with full-scale fabrication, assembly, integration, and test, and that the technical effort is on track to complete system development and operations in order to meet performance requirements within the identified cost and schedule constraints.

Deliverable -A work product that is identified in the Project Plan and is formally delivered to the system owner and other project stakeholders for review and approval.

Dependency -A relationship of one task to another where the start or end date of the second task is related to the start or end date of the first task.

Design - The process of defining the architecture, components, interfaces, and other characteristics of a system, product, or component.

Design specification - A document that describes the design of a software component, product, or system. Typical contents include architecture, control logic, data structures, input/output formats, interface descriptions, and algorithms.

Hardware - Physical computer and other equipment used to process, store, or transmit computer programs or data.

Hierarchy - A structure in which components are ranked into levels of subordination.

Integration Test - Verifies the system components are integrated and working as an application. The technical development team performs this test to uncover errors that occur in the interactions and interfaces between components.

Interface - A shared boundary between two functional units, defined by functional characteristics, common physical interconnection characteristics, signal characteristics, or other characteristics, as appropriate.

Interface requirement - A requirement that specifies an external item with which a software product or system must interact, or that sets forth constraints on formats, timing, or other factors caused by such an interaction.

Lifecycle - See Project Lifecycle.

Life-Cycle Cost - The total cost of ownership over the project's or system's life cycle from design to decommissioning. The total of the direct, indirect, recurring, nonrecurring, and other related expenses incurred, or estimated to be incurred, in the design, development, verification, production, deployment, operation, maintenance, support, and disposal of a project.

Maintenance - The process of supporting a software product or system after delivery to maintain operational status, correct faults, improve performance or other attributes, or adapt to a changed

environment. Maintenance is performed by personnel having specified skill levels, using prescribed procedures and resources, at each prescribed level of maintenance.

Metric - The result of a measurement taken over a period of time that communicates vital information about the status or performance of a system, process, or activity.

Milestone - A scheduled event for which an individual or team is accountable and that is used to measure progress.

Mission - A major activity required to accomplish an Agency goal or to effectively pursue a scientific, technological, or engineering opportunity directly related to an Agency goal. Mission needs are independent of any particular system or technological solution.

Module - A program unit that is discrete and identifiable with respect to compiling, combining with other units, and loading. A logically separable part of a program.

Peer Review - Independent evaluation by internal or external subject matter experts who do not have a vested interest in the work product under review. Peer reviews can be planned, focused reviews conducted on selected work products by the producer's peers to identify defects and issues prior to that work product moving into a milestone review or approval cycle.

Platform - A specific computer and operating system on which a software product or system is developed or operated.

Portability - The ease with which a software component, product, or system can be transferred from one hardware or software environment to another.

Preliminary Design Review (PDR) - A review that demonstrates that the preliminary design meets all system requirements with acceptable risk and within the cost and schedule constraints and establishes the basis for proceeding with detailed design. It will show that the correct design option has been selected, interfaces have been identified, and verification methods have been described.

Procedure - A written description of a course of action to be taken to perform a given task.

Process - An ordered set of steps performed for a given purpose. Processes define or control the development of the project work products. The use of processes will ensure a consistent methodology across all platforms in producing the lifecycle deliverables.

Project - An undertaking of finite duration requiring concerted effort that is focused on developing a specific software product or system.

Project lifecycle - Covers all activities conducted within the scope of an entire project, from project startup to project closeout.

Project Manager - The individual with total responsibility for all activities of a project. The project manager plans, directs, controls, administers, and regulates a project.

Prototyping - A technique for developing and testing a preliminary version of the software product (either as a whole or in modular units) in order to emulate functionality.

Quality assurance - A process designed to provide management with appropriate visibility into the work products being produced and the systems engineering processes being used by the project team.

Reference - A document(s) or other material that is useful in understanding more about an activity.

Regression Test - Re-execution of specific test cases to ensure defects are fixed, find new defects that may have been introduced, and confirm that module(s) are functioning properly.

Reliability - The ability of a software or system component to perform its required functions under stated conditions for a specified period of time.

Requirement - A condition or capability needed by a system owner/user to solve a problem or achieve an objective. A condition or capability that must be met or possessed by the software product or system to satisfy a contract, standard, specification, or other formally imposed documents.

Requirements analysis - The process of analyzing and understanding the scope and feasibility of identified requirements; of developing a preliminary plan to arrive at a detailed definition of system, hardware, or software requirements; and of crystallizing a preliminary system solution.

Requirements Specification - A work product deliverable that specifies the requirements for a software product or system. Typically included are functional requirements, performance requirements, and interface requirements. Describes in detail what will be delivered in the product or system release.

Reusability - The degree to which a software module or other work product or system component can be used in more than one computer program or software system.

Software - Computer programs, procedures, and associated documentation and data pertaining to the operation of a software product or system.

Specification - A document that specifies in a complete, precise, verifiable manner the requirements, design, behavior, and other characteristics of a software component, product, or system.

Stage - A partition of the project lifecycle that divides a project into manageable pieces and represents a meaningful and measurable set of related tasks that are performed to obtain specific work products.

Stakeholder - An individual, group, or organization, who may affect, be affected by, or perceive itself to be affected by a decision, activity, or outcome of a project.

Stakeholder Requirements - Requirements from various stakeholders that will govern the project, including required system capabilities, function, and/or services; quality standards; system constraints; and cost and schedule constraints. Stakeholder requirements may be captured in the Stakeholder Requirements Specification.

Standard - Mandatory requirements employed and enforced to prescribe a disciplined, uniform approach to software and systems development and maintenance.

State Diagram - A diagram that shows the flow in the system in response to varying inputs.

Structured analysis - An analysis technique that uses a graphical language to build models of software products or systems. The four basic features in structured analysis are data flow diagrams, data dictionaries, procedure logic representations, and data store structuring techniques.

System- 1) An integrated set of elements, subsystems, or assemblies that accomplish a define objective. These elements include products (hardware, software, firmware), processes, people, information, techniques, facilities, services, and other support elements. 2) A combination of interacting elements organized to achieve one or more stated purpose.

Systems Analysis - The analytical process by which a need is transformed into a realized, definitive product, able to support compatibility with all physical and functional requirements and support the operational scenarios in terms of reliability, maintainability, supportability, serviceability, and disposability, while maintaining performance and affordability.

System Architecture - The arrangement of elements and subsystems and the allocation of functions to them to meet system requirements.

System Design Document - A work product deliverable that provides a technical description of the solution to meet requirements.

Systems Engineer -An engineer trained and experienced in the field of systems engineering.

Systems Engineering Processes - A logical, systematic set of processes used to accomplish systems engineering tasks.

System of Interest - The system whose life cycle is under consideration.

System & Standards Test – Verifies functional business requirements, business processes, data flows and other system criteria are met.

System Test - Testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

Task - The smallest unit of work subject to management accountability. A task is a well-defined work assignment for one or more project team members. Related tasks are usually grouped to form activities. A task is the lowest level of work division typically included in the Project Plan and Work Breakdown Structure.

Test - The use of system, subsystem, or component operation to obtain detailed data to verify performance or to provide sufficient information to verify performance through further analysis. Testing is the detailed quantifying method of verification and is ultimately required in order to verify the system design.

Testing - An activity in which a software or system component or product is executed under specified conditions, the results are observed and recorded, and an evaluation is made.

Traceability - The degree to which a relationship can be established between two or more products of the development process, especially products having a predecessor-successor relationship to one another.

Transition - The act of delivery or moving of a product from the location where the product has been implemented or integrated, as well as verified and validated, to a customer. This act can include packaging, handling, storing, moving, transporting, installing, and sustainment activities.

Unit - A separately testable element specified in the design of a computer system or software component.

Unit testing - Testing of individual hardware or software units or groups of related units. The isolated testing of each flowpath of code with each unit.

User interface - An interface that enables information to be passed between a user and hardware or software components of a computer system.

User manual - A document that presents the information necessary to use a software product or system to obtain desired results. Typically described are product or component capabilities, limitations, options, permitted inputs, expected outputs, possible error messages, and special instructions.

Validation - The process of evaluating software or systems at the end of the development process to assure compliance with established software and system requirements.

Verification - The process of evaluating a software product or system to determine whether or not the work products of a stage of the project lifecycle fulfill the requirements established during the previous stage.

Walkthrough - An analysis technique in which a team of subject matter experts review a segment of code, documentation, or other work product, ask questions, and make comments about possible errors, violation of development standards, and other problems.

Appendix A - Software Development Plan – Sample Outline

The Software Development Plan describes the developer's plan for conducting a software development effort and gives managers the tools for monitoring the associated processes. It will detail methods and approaches to be followed for all software development activities. It describes all processes and provides guidance for the development team. It will reference standards, methods, tools, actions, reuse strategy, and security responsibilities. The following is a sample outline of a SDP intended for use within the DoD and which can be found at http://www.acqnotes.com/acqnote/careerfields/software-development-plan

- Plan introduction and overview.
    - Purpose, scope, and objectives.
    - Assumptions and constraints.
- Relationship to other program plans.
- Referenced documents.
- Identification of all software and software products to which the SDP applies.
- Definition of terms and acronyms.
- System overview, including system and software architecture.
- Overview of required work, including:
    - Requirements and constraints on the system and software to be developed.
    - Software products and related deliverables.
    - Requirements and constraints on project documentation.
    - The program/acquisition strategy, resources, and schedules
    - Additional requirements and constraints such as on project security, privacy, methods, standards, interdependencies in hardware and software development.
    - Known software-specific risks.
- Charter (Project organization and resources)
- Software-related development processes, approaches, methods, and/or standards including:
    - Overall development methodology.
    - Software Development Standards
    - Software CM Standards
    - Establishing the system/software engineering environment and controls
        - Software Versioning Methodology
        - Software Types/Categories
            - Operational Software (Reusable software products and Commercial off-the-Shelf (COTS))
            - Test Software
            - Support Equipment Software
    - Prototyping and simulations.
    - System requirements analysis and design, including requirements definition and allocation
        - Functional Requirements
        - Operational Requirements
        - Life-Cycle Analysis

        - Computer resources utilization and reserve capacity/growth management.
    - Software requirements analysis.
        - Handling of critical requirements (such as safety, security, and information assurance).
    - Software preliminary and detailed design.
    - Software unit integration and testing.
    - Software component integration and testing.
- Supporting processes and information, including:
    - Software risk management.
    - Approach to requirements traceability.

Appendix B - Digital Services Playbook

The Digital Services Playbook Summary describes thirteen best practices for addressing several key aspects of developing software applications. These described "plays" identify approaches for establishing:

1) clear scope,
2) productive customer experience,
3) unified and simple system layout,
4) process for fast-paced software development,
5) contracts and budget to support modular delivery,
6) life-cycle accountability,
7) experienced government and contract implementation teams,
8) process and environment for utilizing modern technology,
9) flexible infrastructure and platform environments,
10) automation,
11) security through reusable processes,
12) the usage of data to derive value and
13) a path for utilizing open technology.

The beneficial methods found within the Digital Services Playbook will help to reduce cost and time-to-deployment for systems of all sizes and purpose. More details about each of these plays can be found at https://playbook.cio.gov/

“Digital Services” refers to “the delivery of digital information (data or content) and transactional services (e.g., online forms, benefits applications) across a variety of platforms, devices, and delivery mechanisms (e.g., websites, mobile applications, and social media).” Digital services may be delivered to internal customers, external customers, or both.

(Citation)Digital Services Advisory Group, Federal Chief Information Officers Council, and Federal Web Managers Council, "Digital Services Governance Recommendations", Retrieved from https://www.whitehouse.gov/digitalgov/digital-services-governance-recommendations#introduction