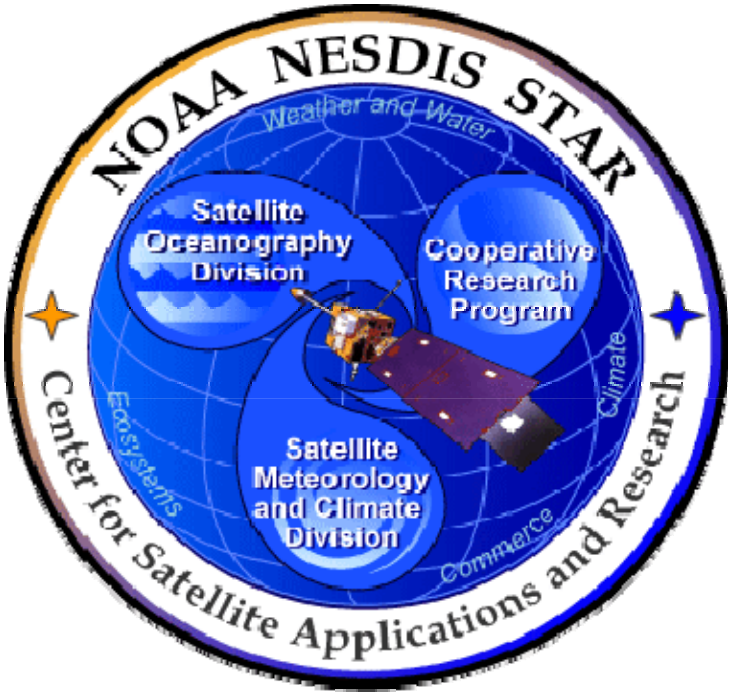

NOAA NESDIS
CENTER for SATELLITE APPLICATIONS and RESEARCH

DOCUMENT GUIDELINE

DG-9.1
UNIT TEST PLAN GUIDELINE

Version 3.0

TITLE: DG-9.1: UNIT TEST PLAN GUIDELINE VERSION 3.0

AUTHORS:

Ken Jensen (Raytheon Information Solutions)

UNIT TEST PLAN GUIDELINE
VERSION HISTORY SUMMARY

Version	Description	Revised Sections	Date
1.0	**No version 1.0 (see Section 1.3 for explanation)**		
2.0	New Document Guideline (DG-12.3) adapted from CMMI guidelines by Ken Jensen (Raytheon Information Solutions)	New Document	10/12/2007
2.1	Minor revisions by Ken Jensen (RIS) for version 2.1.		3/3/2009
3.0	Renamed DG=9.1 and revised by Ken Jensen (RIS) for version 3	1.2, 4.6, 4.13, 4.17, 4.18	10/1/2009

TABLE OF CONTENTS

LIST OF ACRONYMS

CDR	Critical Design Review
CICS	Cooperative Institute for Climate Studies
CIMSS	Cooperative Institute for Meteorological Satellite Studies
CIOSS	Cooperative Institute for Oceanographic Satellite Studies
CIRA	Cooperative Institute for Research in the Atmosphere
CL	Check List
CLI	Check List Item
CM/DM	Configuration Management/Data Management
CMMI	Capability Maturity Model Integration
CREST	Cooperative Remote Sensing and Technology Center
CTR	Code Test Review
DG	Document Guideline
EPL	Enterprise Product Lifecycle
NESDIS	National Environmental Satellite, Data, and Information Service
NOAA	National Oceanic and Atmospheric Administration
PAR	Process Asset Repository
PRR	Project Requirements Review
PBR	Project Baseline Report
PG	Process Guideline
PRG	Peer Review Guideline
QA	Quality Assurance
RAD	Requirements Allocation Document
RAS	Requirements Allocation Sheet
RHTM	Requirements Horizontal Traceability Matrix
RVTM	Requirements Vertical Traceability Matrix
SG	Stakeholder Guideline
STAR	Center for Satellite Applications and Research
SWA	Software Architecture Document
TD	Training Document
TG	Task Guideline
TRR	Test Readiness Review
UTP	Unit Test Plan
VCRM	Verification Cross Reference Matrix
VVP	Verification and Validation Plan

1. INTRODUCTION

The NOAA/NESDIS Center for Satellite Applications and Research (STAR) develops a diverse spectrum of complex, often interrelated, environmental algorithms and software systems. These systems are developed through extensive research programs, and transitioned from research to operations when a sufficient level of maturity and end-user acceptance is achieved. Progress is often iterative, with subsequent deliveries providing additional robustness and functionality. Development and deployment is distributed, involving STAR, the Cooperative Institutes (CICS, CIMSS, CIOSS, CIRA, CREST) distributed throughout the US, multiple support contractors, and NESDIS Operations.

NESDIS/STAR is implementing an increased level of process maturity to support the exchange of these software systems from one location or platform to another. The Unit Test Plan (UTP) is one component of this process.

1.1. Objective

The objective of this Document Guideline (DG) is to provide STAR standards for the UTP. The intended users of this DG are the personnel assigned by the Development Lead to the task of creating a UTP for the project.

1.2. The Unit Test Plan

The UTP contains the test plan for each software unit in the project's product processing system. The UTP, a complement to the project's Verification and Validation Plan (VVP), focuses on the specifics of the software units and the testing of their functionality and performance.

The UTP is produced during the Build phase of the STAR Enterprise Product Lifecycle (EPL)[1].

The initial UTP is UTP v1r0, produced for the Test Readiness Review (TRR)[2]. UTP v1r0 should document the purpose and function of each unit, its traceability to the project requirements, unit data flows, unit components and unit functions to be tested, a test data description, planned test methods and test sequences and identified test risks.

[1] For a description of the STAR EPL, refer to the STAR EPL Process Guidelines (PG-1 and PG-1.A).

[2] Refer to the STAR EPL Process Guidelines (PG-1 and PG-1.A) for a description of the STAR EPL gates and reviews.

Circumstances may occur during unit testing that result in a decision to revise the plan. In that case, the UTP will be updated to v1.1 for presentation at the Code Test Review (CTR).

The intended target audiences are customers, product users, requirements reviewers, code reviewers, code testers, and project managers. The UTP is prepared by the project's development team, under the direction of the Development Lead and in consultation with quality assurance (QA) personnel and the primary customers and users.

The UTP should be developed as a Microsoft Word document. Upon approval, the approved version of the UTP may be converted to an Adobe pdf file for storage in the project artifact repository.

1.3. Background

This DG defines guidelines for producing a UTP. This DG has been adapted from Capability Maturity Model Integration (CMMI) guidelines (CMMI-DEV-v1.2, 2006). It has been tailored to fit the STAR EPL process.

1.4. Benefits

A UTP developed in accordance with the standards in this DG assists the code reviewers in verifying that the pre-operational code satisfies project requirements. It is therefore a requirement that a UTP be developed in accordance with the guidelines in this document.

1.5. Overview

This DG contains the following sections:

Section 1.0	-	Introduction
Section 2.0	-	References
Section 3.0	-	Standard Table of Contents
Section 4.0	-	Section Guidelines
Appendix A	-	Examples
Appendix B	-	Templates

2. REFERENCE DOCUMENTS

SWA: Software Architecture Document contains the software architecture and data flows for the project algorithm. This information will be useful for the developer of the UTP in completing Section 2. This document will be available to approved users in the project artifact repository.

VVP: Verification and Validation Plan is a project artifact that describes the work products to be verified and validated, the requirements for each selected work product and the verification and validation methods for each selected work product. This document will be available to approved users in the project artifact repository.

RAD: Requirements Allocation Document is a project artifact that contains the basic and derived requirements for the work products and the allocation of the requirements to system components and product components. This document will be available to approved users in the project artifact repository.

All of the following references are STAR EPL process assets that are accessible in a STAR EPL Process Asset Repository (PAR) on the STAR web site:

http://www.star.nesdis.noaa.gov/star/EPL_index.php.

PG-1: STAR EPL Process Guideline provides the definitive description of the standard set of processes of the STAR EPL.

PG-1.A: STAR EPL Process Guideline Appendix, an appendix to PG-1, is a Microsoft Excel file that contains the STAR EPL process matrix (Stakeholder/Process Step matrix), listings of the process assets and standard artifacts, descriptions of process gates and reviews, and descriptions of stakeholder roles and functions.

PRG-9: Test Readiness Review Guidelines are the guidelines for the TRR. It is useful for the developer of UTP v1.0 to understand what the reviewers will expect when reviewing the UTP.

CL-9: Test Readiness Review Check List is the check list for the TRR. It is useful for the developer of UTP v1.0 to understand the specific Check List Items (CLI) that the reviewers of the UTP will be required to approve.

PRG-10: Code Test Review Guidelines are the guidelines for the CTR. It is useful for the developer of UTP v1.1 to understand what the reviewers will expect when reviewing the UTP.

CL-10: Code Test Review Check List is the check list for the CTR. It is useful for the developer of UTP v1.1 to understand the specific CLI that the reviewers of the UTP will be required to approve.

DG-0.1: STAR Document Style Guideline is a STAR EPL Document Guideline (DG) that provides STAR standards for the style and appearance of STAR documents developed as Microsoft Word files

SG-14: STAR EPL Development Scientist Guidelines provides a description of standard tasks for Development Scientists, including development of the UTP.

SG-15: STAR EPL Development Tester Guidelines provides a description of standard tasks for Development Testers, including development of the UTP.

SG-16: STAR EPL Development Programmer Guidelines provides a description of standard tasks for Development Programmers, including development of the UTP.

TG-9: STAR EPL Code Development and Test Task Guidelines provides a description of standard tasks for process step 9, during which UTP v1.0 is developed.

TG-10: STAR EPL Code Test and Refinement Task Guidelines provides a description of standard tasks for process step 10, during which UTP v1.1 is developed.

TD-9: Project Requirements is a STAR EPL Training Document (TD) that provides a description of techniques and tools for requirements development and management.

3. STANDARD TABLE OF CONTENTS

4. SECTION GUIDELINES

This section contains the STAR guidelines for each section of the UTP.

The UTP should follow the STAR standard for style and appearance, as stated in DG-0.1.

4.1. Table of Contents

The Table of Contents can be inserted by using Word's Insert → Reference → Index and Tables → Table of Contents function or by pasting the Table of Contents from this DG into your document and updating it for the section headers you make for your document. Use a page break if necessary to ensure that the Table of Contents appears at the top of a page.

4.2. List of Figures

A List of Figures should be provided after the Table of Contents. A page break should be used if necessary to ensure that the List of Figures appears at the top of a page. To create a List of Figures, use Word's Insert → Reference → Index and Tables → Table of Figures function, selecting the "Table of Figures" Style. Alternatively, the List of Figures can be created by pasting the List of Figures for this DG into your document.

Figures should be created by using Word's Insert → Picture → From File function or Word's Insert → Object function. Figures should be numbered X.Y, where X is the main section number where the figure resides and Y = 1,N is the ordered number of the figure in the section. Figure captions should have Arial bold 12 point font, should be center justified, and should have a "Table of Figures" Style. A Figure Caption template is provided in Appendix B of this DG.

4.3. List of Tables

A List of Tables should be provided after the List of Figures. The List of Tables can appear on the same page as the List of Figures, with three blank lines separating them, provided both lists can fit on the same page. If both lists cannot fit on the same page, a page break should be used to ensure that the List of Tables appears at the top of a page.

To create a List of Tables, use Word's Insert → Reference → Index and Tables → Table of Figures function, selecting the "Table - Header" Style. Alternatively, the List of Tables can be created by pasting the List of Tables for this DG into your document.

Tables should be created with the Table → Insert → Table function. Tables should be numbered X.Y, where X is the main section number where the table resides and Y = 1,N is the ordered number of the table in the section. Table titles should have Arial bold 12 point font, should be center justified, and should have a "Table - Header" Style. A Table Title template is provided in Appendix B of this DG. Table text should have Arial regular 10 point font.

4.4. List of Acronyms

The use of acronyms is encouraged. A two word or longer name for an item (e.g., Unit Test Plan) should be given an acronym (e.g., UTP) if the name is used more than once in the document. A List of Acronyms should be provided after the List of Tables. The List of Acronyms should be in alphanumeric order. Use the List of Acronyms in this DG as a template. A page break should be used if necessary to ensure that the List of Acronyms appears at the top of a page.

4.5. Section 1 – Introduction

The UTP shall include an Introduction Section. This section shall include

- A well-defined purpose and function for the document
- Specific intended user(s)
- How the intended user(s) should use the document
- A responsible entity for generating the document
- A responsible entity for review/approval of the document
- A responsible entity for storage, accessibility, and dissemination
- A brief overview of the contents of each main section

4.6. Section 2 – Software Units

This is a brief section whose purpose is to list the product processing system's software units whose test plans will be described in the sections to follow. The software units are the

primary components of the product processing software architecture subset of the system architecture. They are documented in the project's Software Architecture Document (SWA). This section should only contain the list of software units and a System-Layer process flow diagram, also known as a Layer-1 data flow diagram. Each software unit that will be tested should be highlighted in the diagram. The list of software units and the Layer-1 data flow diagram should be adopted without revision from the SWA.

Each software unit should have a number and a name, as they are documented in the SWA.

Each unit test is described in its own section. Section 3 is reserved for the first unit test, Section 4 for the second unit test, etc. It is recommended that the order of the units in the UTP matches the sequential order of the processing system. This order should be reflected in numerical identifications assigned to the units in the system architecture, as documented in the SWA and listed here.

The guidelines for each unit's section (Section 3, 4, etc., of the UTP) are identical. Each section should contain the following subsections:

1) Purpose and Function
2) Process Flow
3) Test Items
4) Requirements Trace
5) Test Data Description
6) Test Environment
7) Test Configuration Build
8) Test Methods
9) Test Sequence
10) Unit Test Risks
11) Expected Unit Test Results
12) Unit Test Success Criteria

The following guidelines are provided for subsections 3.1 - 3.12. It is understood that these apply to every unit's section (Section 4, 5, etc.).

4.7. Section 3.1 – Purpose and Function

Explain the purpose of the unit, the role of the unit in the product processing system, the major functional steps, and how these steps satisfy the purpose of the unit. The content should be primarily textual. References to appropriate figures and tables in SWA v2r1 should be made. This explanation should be consistent with, and can be obtained from, the Unit Detailed Design Document (Section 2.1).

4.8. Section 3.2 – Process Flow

Show the unit process flow as a process flow diagram, also known as a Layer-2 data flow diagram. This diagram should be adopted without revision from the SWA.

4.9. Section 3.3 – Test Items

Identify all unit components that have been selected for testing. These items should be part of the system architecture as documented in SWA v2r1. Typically, they are sub-processes of the unit's process flow that is shown in Section 3.2. They should also be identified as verification items in the project's VVP. It is important that the development team confirm that the planned unit test items are documented as verification items in the VVP. Discrepancies must be resolved and either the UTP or the VVP revised as necessary.

It is useful to number each test item, so that they may be referred to by their number in later subsections of the UTP. Test items should be numbered as they are numbered in SWA v2r1. If there are test items that are not identified in the SWA, note these and give them appropriate numbers.

4.10. Section 3.4 – Requirements Trace

Identify the requirements allocated to each test item that is listed in Section 3.3, consistent with the requirements allocation documented in the Requirements Allocation Document (RAD). This is essential, as it is the requirements for the test items that are to be verified. Without requirements and requirements allocation, there is no basis for the unit test. They should be consistently documented in the VVP and the RAD.

Requirements include code functionality and product performance. To trace the test items to their driving requirements, refer to the RAD. The RAD will include several matrices that will be useful for identifying these requirements, including a Requirements Allocation Sheet

(RAS), a Requirements Vertical Traceability Matrix (RVTM), and a Requirements Horizontal Traceability Matrix (RHTM). Refer to TD-9 for a description of the RAS (Section 6.4), the RVTM (Section 7.1.1), and the RHTM (Section 7.1.2).

Each requirement is identified in the RAD with a unique identification number and a requirements statement. Basic requirements will be numbered 1.0, 2.0, 3.0, etc. Requirements derived from basic requirement 1.0 will be numbered 1.1, 1.2, 1.3, etc. Requirements derived from derived requirement 1.3 will be numbered 1.3.1, 1.3.2, 1.3.3, etc. Use the unique identification numbers for the requirements trace.

The following format is recommended for this subsection:

Test Item 1

{Requirement Number – Requirement Statement (first requirement)
Requirement Number – Requirement Statement (second requirement)
Requirement Number – Requirement Statement (third requirement)
etc.}

Test Item 2

{Requirement Number – Requirement Statement (first requirement)
Requirement Number – Requirement Statement (second requirement)
Requirement Number – Requirement Statement (third requirement)
etc.}

Test Item 3 (etc.)

where the set of requirements between the braces is listed in numerical order:

{1.0
1.1
1.1.1, etc.
1.2
1.2.1, etc.
2.0
2.1
etc.}

Following this convention allows the reviewers to most easily compare the requirements trace documented in this subsection with the RAD.

4.11. Section 3.5 – Test Data Description

List all data files that will be used as input files for the unit test. “Test data” includes sensor data (real, proxy, or simulated), ancillary data, control files, parameter files, and look up tables. It is recommended that these be listed in a table. Table 5.1 provides an example.

Table 5.1 – Input Test Data Items

Test Data Item	Type	Filename	Design Description Filename
1	Real sensor data	IASI_L1_Thin_001.bin	IASI L1 thinned radiances
2	Ancillary data	AVHRR_CM_001.bin	AVHRR cloud mask
3	Parameter File	IASI_AVTP.par	AVTP parameter file
4	Look up table	AVTP.lut	AVTP look up table

The table should include every input file in the design description, as documented in the unit’s DDD. Following the table, describe each input data file in sufficient detail for a reviewer to be able to confirm that its contents and format matches the description of the appropriate input file documented in Section 4 of the unit’s DDD. The DDD will probably have a generic file identification (e.g., “IASI Level 1 thinned radiances”), but your description will be specific (e.g., “IASI_L1_Thin_092307.08.41”). Make sure you identify your specific filename with the generic file identification in the DDD.

It is recommended that one of the following two formats be used for the description:

Test Data Item 1 - <Description of item 1>

Test Data Item 2 - <Description of item 2>

Test Data Item 3 - <Description of item 3>, etc.

- OR -

1) <Description of item 1>

2) <Description of item 2>

3) <Description of item 3>, etc.

List all “truth” data sets that will be used to assess the performance of the unit test. Truth data sets typically contain the values of environmental or weather products that are traceable to performance requirements (e.g., Atmospheric Vertical Temperature profiles). Truth data sets may be real, proxy or simulated data. Explain how each real or proxy truth data set has been obtained. Explain how each simulated truth data set has been constructed.

4.12. Section 3.6 – Test Environment

Describe the environment in which the unit test will be performed. Demonstrate that the planned test environment complies with the project’s test environment requirements, as documented in the RAD.

A project may use the development environment to test the pre-operational system or it may choose to establish a separate test environment. Project constraints will usually determine this choice. For example, operations may request that the test environment be a clone of the operational environment, but cost factors may exclude establishing the development environment as an operational clone. In that case, the best solution may be to use a small operational clone environment as a separate test environment. A project may also choose to perform its pre-operational code unit tests in the development environment and then perform its system integration tests in an operational clone environment. In any case, these choices should be explicitly stated as requirements for the test environment.

Test environment requirements and test plans should be developed iteratively, with communication between the developers. Typically, preliminary test environment requirements are established at the Project Requirements Review (PRR), driven by STAR standards and operational needs. There may be cases where a project’s development team determines that the preliminary test environment requirements are not ideal for their project. If this occurs, the development team can request an analysis and possible refinement of the test environment requirements. Nevertheless, the unit test plan must comply with the approved test environment requirements at the time of the unit test.

For most projects, the same test environment will be used for all unit tests. It is rare, possibly inconceivable, that separate test environments would be established for different unit tests of the same project. If your project will use the same test environment for all of its unit tests, it is sufficient for the comparable subsections for the remaining unit tests (section 4.6, 5.6, etc.) to simply state that the test environment is identical to that described in Section 3.6.

4.13. Section 3.7 – Test Configuration Build

Identify all configuration items that will be used in the unit test, including code modules, test data sets, utilities, libraries, etc. The set of configuration items is called the test configuration. Each item in the test configuration must be included in the project baseline under configuration control and should be documented in the Project Baseline Report (PBR). Work with STAR Configuration Management/Data Management (CM/DM) to ensure that the necessary items are added to the baseline and documented in the PBR for TRR. This is essential to confirm test readiness.

Identify the procedure to build the test configuration into a test executable. This is necessary to prepare for the unit tests.

4.14. Section 3.8 – Test Methods

Describe the methods that will be used to test each test item. Test methods should be closely related to the verification methods that are documented in the VVP. The standard methods include Analysis, Demonstration, Inspection, and Test. Note that unit test methods are not restricted to “Test”. The “Test” verification method refers specifically to a procedure to quantitatively demonstrate compliance with performance specifications. Although unit test methods will often include “Test” methods for verifying quantitative requirements, they can also include, and usually will include, other methods for verifying other requirements. Note which test items will be verified with each method or combination of methods.

Demonstrate that the methods selected for verification of a given item will address the requirements to be verified for that item. The project’s VVP should contain material that can be used for this subsection of the UTP, including a Verification Cross Reference Matrix (VCRM) that relates each Verification Item to the method or methods planned for its verification. It is permissible to insert relevant material from the VVP into the UTP, provided it is referenced appropriately. Alternatively, the UTP developers may choose to leave the

specific material out of the UTP and refer to specific sections of the VVP that pertain to the test methods for this unit. The latter choice is recommended if the TRR and CTR reviewers are already familiar with the material in the project's VVP. It is recommended that the UTP developers consult with the TRR and CTR reviewers before deciding how to fill this subsection of the UTP.

4.15. Section 3.9 – Test Sequence

Describe the planned sequence of test actions in sufficient detail that a reviewer can confirm that all test items are exercised, all test data are utilized, all planned test methods are used as planned, and the planned output will allow a reviewer to confirm that the requirements identified in Section 3.4 will be satisfied. It is recommended that this subsection assist the reviewer by specifically noting which sequence steps exercise which test items, utilize which test data sets, and use which test methods. It is recommended that this be captured in a matrix whose rows contain the test actions and whose columns contain the test items, test data sets and test methods. If the matrix is too large, consider using separate matrices for test items, test data sets and test methods.

It is recommended that diagnostic code be added to the unit code to facilitate the verification of functionality and outputs. If this is done, two versions of the unit code should be created. The first version will contain the diagnostic code. The second version, with all diagnostic code stripped out, is the baseline pre-operational code. The unit test sequence should then include the execution of both code versions. The unit code with diagnostic code should be run first, with all diagnostic messages and outputs captured for verification (c.f., Section 5.17 of this DG). This provides a complete verification of code functionality and performance. Then, the baseline pre-operational unit code should be run. Verification of the baseline code is then achieved by confirming that its outputs are identical to the outputs from the unit code with diagnostic code. The test sequence should include this final verification step.

4.16. Section 3.10 – Risks

Identify and evaluate risks to successful implementation of the test plan for this unit. Verification risks for the project will be documented in the project's VVP. Some of these may pertain to this unit test. Conversely, new risks identified for this subsection should be included in the VVP. The UTP and VVP developers should consult with each other to

ensure that the latest risk assessment is consistently described in each document. STAR EPL Process Guideline PG-1 explains how to evaluate verification risks.

4.17. Section 3.11 – Expected Unit Test Results

Describe the expected output from each sequence step that is identified in Section 3.9. The expected output should be described in sufficient detail for unit test reviewers to be able to confirm that the actual unit test output matches the expected output. Output includes runtime messages, diagnostic messages and the content of data files.

Runtime messages are messages written by the operating system to a runtime log file or other designated output source (e.g., a monitor connected to the computer from which the program execution command has been entered). These may occur if the unit code is written to generate such messages as a way to test functionality. Describe the exact content of each expected runtime message and at what point in the test sequence it is expected.

Diagnostic messages are messages written by the unit program to a runtime log file or other designated output source (e.g., a monitor connected to the computer from which the program execution command has been entered). The nominal purpose of a diagnostic message is to report a functional result (e.g., ‘subroutine X called’) or the quantitative value of an input, intermediate, or output variable (e.g., ‘X(50) = 7’). Describe the exact content of each expected diagnostic message and at what point in the test sequence it is expected.

Data files include the output data sets that are designed to be produced by the unit program. In addition, a diagnostic program may write intermediate data sets to diagnostic files. Describe the expected content of all diagnostic files that will be sampled as part of the unit test.

Note expected run times. Demonstrate that run times meet operational run time requirements. This demonstration can include scaling factors if the test environment is predicted to be slower than the operational environment by a known amount.

4.18. Section 3.12 – Unit Test Success Criteria

State the success criteria for the unit test. Success criteria should include the following:

- All test sequence steps run as planned in the UTP

- Program run time meets requirements
- All runtime message and diagnostic messages are written as expected in the UTP
- The contents of all diagnostic files are as expected in the UTP
- Unit test output satisfies all quantitative performance requirements

4.19. Section 4 – List of References

This section should consist of a list of all references cited in the document. Include all references deemed useful by the Product Team. References should be listed in alphabetical order. References that begin with an author list should begin with the last name of the lead author. A template is provided in Appendix B.

APPENDIX A - EXAMPLES

An example of a UTP that follows the STAR standards and guidelines will be developed and placed in the STAR EPL PAR.

APPENDIX B - TEMPLATES

This appendix contains templates for specific pages and sections of the UTP.

B.1 Cover Page Template:

In this template, <X> = 1.0 for version 1, <X> = 1.1 for version 1 revision 1, <X> = 2.0 for version 2 etc. <Project Name> should be the actual approved name of the Project.

NOAA NESDIS
CENTER for SATELLITE APPLICATIONS and RESEARCH

<PROJECT NAME>
UNIT TEST PLAN

Version <X>

B.2 Document Header Template:

In this template, <X> = 1.0 for version 1, <X> = 1.1 for version 1 revision 1, <X> = 2.0 for version 2 etc.

In this template, <Project Name> should be the actual approved name of the Project.

In this template, <Y> = the actual page number.

In this template, <Z> = the actual total number of pages

NOAA/NESDIS/STAR

UNIT TEST PLAN
Version: <X>
Date: <Date of Latest Signature Approval>

<Project Name>
Unit Test Plan

Page <Y> of <Z>

B.3 Document Cover Page Footer Template:

Hardcopy Uncontrolled

B.4 Document Footer Template:

Hardcopy Uncontrolled

B.5 Approval Page Template:

In this template, <X> = 1.0 for version 1, <X> = 1.1 for version 1 revision 1, <X> = 2.0 for version 2 etc. <Project Name> should be the actual approved name of the Project.

TITLE: <PROJECT NAME> UNIT TEST PLAN VERSION <X>

AUTHORS:

<Lead Author>

<Co-Author 1>

<Co-Author 2>

<etc.>

APPROVAL SIGNATURES:

_____ <Actual Signature Date>
<Name of Project Development Lead> Date
Project Development Lead

_____ <Actual Signature Date>
<Name of Project Manager> Date
Project Manager

_____ <Actual Signature Date>
<Name of Agency Approver> Date
Agency

B.6 Version History Page Template:

In this template, <Project Name> should be the actual approved name of the Project.

<PROJECT NAME>
UNIT TEST PLAN
VERSION HISTORY SUMMARY

Version	Description	Revised Sections	Date
1.0	Created by <Name of Developer(s)> of <Name of Developers' Agency/Company> for Test Readiness Review.	New Document	<Actual date of Latest approval signature>
1.1	Revised by <Name of Developer(s)> of <Name of Developers' Agency/Company> for Code Test Review	<applicable sections>	<Actual date of Latest approval signature>
etc.			

B.7 Figure Caption Template:

Figure X.Y - <Figure caption in Arial regular 12 point font>

B.8 Table Title Template:

Table X.Y - <Table title in Arial regular 12 point font>

B.9 List of References Template:

Ackerman, S. (1997). Discriminating clear-sky from cloud with MODIS: Algorithm Theoretical Basis Document, Version 3.2.

Asrar, G., M. Fuchs, E. T. Kanemasu, and J. L. Hatfield (1984). Estimating absorbed photosynthetically active radiation and leaf area index from spectral reflectance in wheat. *Agron. J.*, 76:300-306.

Bauer, E., and Kohavi, R., (1998). An empirical comparison of voting classification algorithms: bagging, boosting, and variants, *Machine Learning*, **5**: 1-38.

Bonan, G.B. (1995). Land-atmosphere interactions for climate system models: Coupling biophysical, biogeochemical, and ecosystem dynamical processes. *Remote Sens. Environ.*, 51:57-73.

Friedl, M. A., and C.E. Brodley (1997). Decision tree classification of land cover from remotely sensed data. *Remote Sens. Environ.*, 61:399-409.

Scepan, J. (1999), Thematic validation of high-resolution global land-cover data sets. *Photogramm. Eng. Remote Sens.*, 65:1051-1060.

Shukla, J., C. Nobre, and P. Sellers (1990). Amazon deforestation and climate change. *Science*, 247:1322-1325.

Wilson, M.F., and A. Henderson-Sellers (1985). A global archive of land cover and soils data for use in general circulation models. *J. Clim*., 5:119-143.

Wu, A., Z. Li, and J. Cihlar (1995). Effects of land cover type and greenness on advanced very high resolution radiometer bidirectional reflectances: analysis and removal. *J. Geophys. Res.*, 100: 9179-9192.

END OF DOCUMENT