Software Quality Assurance Plan

Submitted to:

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Marshall Space Flight Center, AL 35812

Submitted by:

Center for Space Research
Massachusetts Institute of Technology
Cambridge, MA 02139

Document No.36-01215-01
Contract # NASA-37716
June 23, 1994

Approvals:

Edward A. Boughan Project Engineer Massachusetts Institute of Technology	Philip J. Gray Project Manager Massachusetts Institute of Technology

---

Table of Contents

Acronyms & Abbreviations

ACIS	AXAF CCD Imaging Spectrometer
AXAF	Advanced X-ray Astrophysics Facility
CCB	Configuration Control Board
CEI	Contract End Item
CEI	Contract End Item
CMP	Configuration Management Plan
CSR	Center for Space Research
DPA	Digital Processing Assembly
DR	Document Requirements
ECO	Engineering Change Order
EGSE	Electronic Ground Support Equipment
ICD	Interface Controlled Description
MIT	Massachusetts Institute of Technology
MSFC	Marshall Space Flight Center
PAM	Product Assurance Manager
PDR	Preliminary Design Review
PE	ACIS Project Engineer
PECO	Proposed ECO
PM	ACIS Project Management
PRR	Program Requirements Review
RCS	Revision Control System
ROM	Read Only Memory
S/C	Spacecraft
SDP	ACIS Software Development Plan
SIS	Science Instrument Software
SMP	Software Management Plan
SPM	Software Project Management
SPR	Software Problem Report
SQA	Software Quality Assurance
SQAP	ACIS Software Quality Assurance Plan
SSS	Software Sub-Section
STP	ACIS Software Test Plan

```
TBD         To Be Determined
```

1. Introduction

1.1 Purpose

The purpose of this Software Quality Assurance Plan (SQAP) is to define the techniques, procedures, and methodologies that will be used at the Center for Space Research (CSR) to assure timely delivery of the software that meets specified requirements within project resources. This document is written in response to Section 2.2.3 of the Marshall Space Flight Center (MSFC) Software Management and Development Requirements Manual (MM-8075.1). The format of this plan follows the requirements found in the tailored MSFC Software Quality Assurance Requirements for MSFC Projects (CQ-5530.1A) document.

1.2 Scope

The use of this plan will help assure the following: (1) That software development, evaluation and acceptance standards are developed, documented and followed. (2) That the results of software quality reviews and audits will be given to appropriate management within CSR. This provides feedback as to how well the development effort is conforming to various CSR development standards. (3) That test results adhere to acceptance standards.

2. Applicability

The processes described in this plan are used during the requirements, design, implementation, test, verification and acceptance of the ACIS instrument. ACIS contains a Digital Processing Assembly (DPA) which contains the Science Instrument Software (SIS) The DPA controls instrument operation, extracts valid X-ray events, and processes the data for downlink. It is anticipated that the SIS will contain multiple Software Sub-Sections (SSS) These items are identified in the ACIS Software Development Schedule (TBD). This plan is in effect until the DPA portion of the ACIS instrument is accepted by MSFC.
It is anticipated that certain subsections of the Electronic Ground Support Equipment (EGSE) will be used to calibrate and validate various aspects of the ACIS instrument. The development of the software contained in these subsections of the EGSE follow the processes contained in the AXAF-I Software Management Plan. There will be no quality assurance activities for this equipment. When the PE deams appropriate, there may be some testing functions performed by SQA.

3. Applicable Documents

The following documents can be used as requirements for the design and manufacture of ACIS software and form a part of this document to the extent specified herein. The issue in effect at the time of contract award shall apply unless otherwise listed below.

3.1 NASA Documents

MM-8075.1

MSFC Software Management and Development requirements Manual (Jan. 22, 1991)

CQ-5330.1A

MSFC Software Quality Assurance Requirements for MSFC Projects (Rev A)

NHB-8070.5

NASA Requirements for Significant Problem Reporting and Trend Analysis (Apr 88)

NMI-8070.3B

NASA Problem Reporting, Corrective Action, and trend Analysis Requirements (Apr 93)

3.2 ACIS Documents

ACIS-36-01201 (Section o)

ACIS Program Management Plan: Instrument Software Management Plan (SMP)

ACIS-36-01201 (Section p)

ACIS Program Management Plan: Instrument Software Development Plan (SDP)

ACIS-36-01101

ACIS CEI Specification

ACIS-36-

ACIS Software Test Plan (STP)

ACIS-36-01103

ACIS Software Requirements Specification

ACIS-36-01206

ACIS Configuration Management Plan (CMP)

4. Program Management and Planning

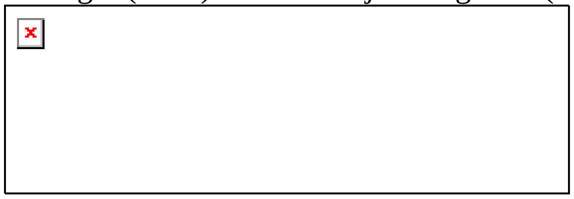
The technical and managerial process necessary for satisfying software related project requirements are described in the ACIS Instrument Software Management Plan (SMP).

4.1 The ACIS SQA Plan

Throughout this plan, the software development effort controlled by the Center for Space Research (CSR) at the Massachusetts of Technology (MIT) for ACIS development is identified as CSR/ACIS. Except for commercially available, of the shelf software, the entire software development effort will occur within CSR at MIT which is considered a single site. This plan is submitted to MSFC for comment following the ACIS software audit in April of 94. This plan can be modified and controlled as specified in the ACIS Configuration Management Plan (CMP).

4.2 Organization

The organization of the ACIS project at CSR consists of three branches. This organization is shown in Figure 1. The science branch is headed by a CO Principal Investigator. The engineering branch is headed by the ACIS Project Manager (PM). ACIS development is the responsibility of the ACIS Project Engineer (PE) with interface issues involving other institutions coordinated by the ACIS System Engineer (SE). The Performance Assurance branch is headed by the Performance Assurance Manager (PAM) with assistance from a Software Quality Assurance (SQA) engineer. The software development effort is being managed by the Software Project Manager (SPM) and the Project Engineer (PE).



sqap.r1.figure.id.5.gif

Figure 1: ACIS Management Structure

Certain members of the quality function have two reporting functions. The PAM reports directly to the Director of CSR with a heavy dotted line to the PM. This means that the PAM administratively reports to the Director of CSR while daily activities are coordinated with the PM. For software development, the SQA engineer administratively reports to the PAM while daily activities are coordinated by the Software Project Engineer. (SPE) This organization allows for uninhibited objective evaluations and recommendations while allowing for prompt corrective action.

4.3 Tasks

All software development tasks for ACIS, within the scope of this document, will be performed by personnel in the engineering branch. The development of the Science Instrument Software (SIS) is managed by the SPM. The development of the software in the EGSE is managed by the PE. SQA will assure that the SIS in the DPA is validated and that internal as well as external deliverables are produced in accordance with the SDP, SMP, STP and this document. SQA will certify that the Science Instrument Software (SIS) acceptance process used in the delivery of the DPA and will verify the acceptability for use of specific Electronic Ground Support Equipment (EGSE) used to validate the delivered ACIS Instrument.

The PAM will assure that the SQA function is being performed according to this plan.

4.4 Software Training

4.4.1 SQA Personnel

No training of SQA personnel is anticipated as the person tasked is the author of the SQA plan and he also reviews the ACIS plans identified in the reference documents section of this plan. SQA qualifications are documented on respective resumes. In the event that new SQA personnel are required, the PAM will ensure that they are familiar with the contents and intent of this plan.

4.4.2 Software Developer Training Certification

Each member of the software development team will be given a copy of the SDP and SMP. SQA ensures this by interviewing each software developer within one month of their beginning to work on the ACIS project. A log will be kept by SQA to record the interview.

---

5. SQA Program Requirements

This section defines the SQA review, reporting, and auditing procedures used at CSR/ACIS to ensure that internal and external software deliverables are developed in accordance with this plan and contract requirements. Internal deliverables are items that are produced by software development and then filed using configuration control procedures. e.g. requirements specifications, software architecture descriptions, scenario descriptions, executable code, release notes. External deliverables are items that are produced for organizations outside of MIT. e.g. user manuals TBD.

5.1 Program Resources Allocation Monitoring

Computer resource (memory and timing) allocation and utilization of target hardware is updated by Software Project Management and reported monthly to ACIS Project Management. SQA will audit the monthly status reports to ensure that computer resources utilization is included.

5.2 SQA Program Audits

In order for SQA to evaluate compliance with the CSR/ACIS SDP, SMP, and the STP, SQA will review and approve the above plans. These plans will specify that the evidence of work generated is adequate to ensure compliance with project and contract requirements. Audits performed shall include examination of both internal and external software deliverables.

5.2.1 Scheduled Audits

SQA will generate and maintain an Audit Schedule. Audits will occur at the end of each development phase as indicated in the SMP. The results of audits will be discussed with the individual most responsible for the production of the deliverable.

5.2.2 Unscheduled Audits

SQA will perform random and unannounced audits to ensure the corrective actions agreed to during the Scheduled Audits are being followed. The results of audits will be discussed with the individual most responsible for the production of the deliverable.

5.2.3 Audits of the SQA Organization

The SQA function will be audited by the PAM prior to any MSFC project review in which software is involved. The results of the audit will be available to MSFC at the project review. Checklists will be used.

5.2.4 Audit Reports

Audit reports, and recommended corrective actions generated by SQA will be brought to the attention of the individual most responsible for producing the software deliverable. Corrective action will be recommended and reviewed with the individual and SPM.
The results of audits of the SQA function will be kept by the PAM.

5.3 SQA Records

Audit reports contain the recommended actions for correction or improvement. Copies of scheduled and unscheduled audits and reviews will be kept by SQA. Audit, change, interview, and review logs will be kept by SQA. After the applicable; ECO has been completed, Software Release folders containing a copy of the released ECO and Release Notes will be kept by SQA.

5.4 SQA Status Reports

SQA status is reported monthly to MSFC. The content of this report will identify: (1) items produced by the SQA functions; (2) significant software development compliance problems, if any, along with their agreed to recommended corrective actions. (3) audits, reviews, and tests accomplished during the reporting period.

5.5 Software Documentation

SQA will review all MSFC deliverable software documentation including software plans. Review checklists will be used to review these documents. These reviews will help ensure that documentation is in compliance with applicable DRs and CSR/ACIS generated plans and procedures.
In general, the essential software documentation should include: (1) Software requirements; (2) user documentation, (3) analysis documentation that captures semantics of the system's function points as viewed through scenarios (4) architectural and implementation documentation that communicates the vision and details of the architecture to the development team. In general, the essential documentation of a system's architecture and implementation should include the following:

- A description of the high level architecture,
- A description of the key abstractions and mechanisms in the architecture,
- A description of the scenarios that illustrate the as-built behavior of key aspects of the system,

Software documentation must be based on some published convention such as found in IEEE Software Engineering Standards This convention will be specified in the SDP.
Released documents will be audited to ensure that published conventions were followed and that appropriate review comments were incorporated. PECOs will be generated by SQA in accordance with the CMP if discrepancies are found. A log of SQA generated PECOs and subsequent ECOs will be maintained by SQA to ensure closure.
Software source code commenting requirements are described in Attachment 1.

5.6 Requirements Traceability

Traceability is identified through the use of a spreadsheet matrix which will tie individual Contract End Item (CEI) Specifications, applicable Interface Controlled Description (ICD)s and Software Requirements Document entries to lower level design and test specification paragraphs or sections. These Traceability products are produced and maintained by SQA.

SQA is included in review process for all software document generation. During these reviews, checklists and Traceability spreadsheets are used to ensure that requirements are met by both the design and test functions.

5.7 Software Development Process

SQA will audit deliverables between each software development life cycle phase until the software or subsystem is permanently incorporated in to the DPA hardware. The software life cycle phases will be defined in the SMP and the software products will be identified in the Software Development Schedule.. The SMP and SDP will not preclude SQA from performing unannounced audits at any time during the software development cycle. A software development folder or equivalent, as identified in the SMP will be used to hold or point to appropriate design, implementation, test, and validation documentation for each appropriate software module or function.

5.8 Project reviews

5.8.1 Formal Reviews

At least one week prior to delivery of documents to MSFC for a formal review, SQA will review the Document List that is generated by the SPE. This list identifies all the documents and revision that will be submitted for the formal review. SQA will review software related documents identified on this list to ensure that the indicated revision is or will be available in time for shipment to MSFC. Discrepancies will be brought to the attention of the SPM.

5.8.2 Informal Reviews

5.8.2.1 Design Walk-throughs

SQA will be invited to all design walk-throughs throughout the entire software development life-cycle to ensure that peer and management reviews of the software design are conducted. The SPM will ensure that a verifiable process is used to identify all action items generated during this review process. SQA will audit this process to ensure that all action items have been addressed.

5.8.2.2 Code Walk-throughs

SQA will be invited to all code walk-throughs to ensure that peer reviews of the source code are conducted. The SPM will ensure that a verifiable process is used to identify all action items generated during this review process. SQA may then audit this process to ensure that all action items have been addressed.

5.8.2.3 Baseline Quality Reviews

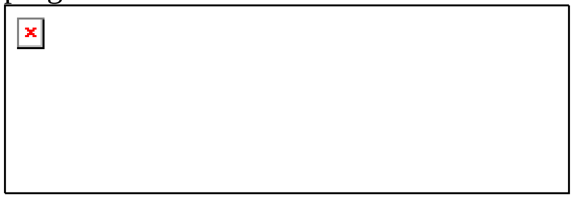
These reviews will be conducted by SQA prior to any baseline release of executable code that is identified with an alphabetic revision ID. This review ensures that: (1) the code has been tested and meets module specifications, except as noted; (2) that any changes to applicable software module design documents have been identified; (3) that appropriate validation tests have been run; (4) that the functionality of the baseline is documented. (5) that all software design documentation complies with this plan and other applicable CSR/ACIS plans and procedures. (6) that tool and techniques used to produce and validate the Software Sub-System are identified and controlled.

5.8.2.4 Configuration Audits

Compliance to this plan will be assured by SQA auditing software ECOs to ensure the following: that the ECO form is complete and has the appropriate signatures; that the ACIS Configuration Data Base is updated; that the item(s) under revision as indicated in the ECO are identified as specified. A software ECO can be identified by inspecting the ECO for a change to a software part with an identification number of 36-?xxxx. In all cases, the documentation of these spot audits will be maintained in the office of the SQA Engineer.

5.9 Tools and Techniques

SQA will assure that all purchased or developed tools that directly effect the contents of the software that resides in the DPA are uniquely identified, tested/evaluated. For example, compilers, linkers, loaders, boot transfer programs, checksum generation programs fall under this category. These tools and programs are identified in the SDP. Text editors, Case tools, unit test programs need not be controlled.



The assurance process occurs during the Baseline Quality Review mentioned above.
sqap.r1.figure.id.6.gif
Figure 2 Configuration with the use of SW Revision Control System

5.10 Software Configuration Management

Software configuration management is the progressive, controlled definition of the shape and form of the software deliverables. It integrates the technical and administrative actions of identifying, documenting, changing, controlling, and recording the functional characteristics of a software product throughout it's life cycle. It also controls changes proposed to these characteristics. As the software product proceeds through requirements, analysis, implementation, test, and acceptance, the identification of programs are identified in the SDP. This assurance process occurs during the Baseline Quality Review mentioned above. its configuration becomes progressively more definite, precise, and controlled. Software configuration management is an integral part of the project configuration management, using the same procedures for identification and change control that are used for documents, engineering drawings, test verification procedures, etc. The SPE will ensure that SQA is part.
Referring to Figure 2, the purpose of the software development process is to produce software that satisfy science and interface requirements. As part of the development process, source, documentation and make files are also produced. All these files, except for the executable, are stored using a Revision Control System. A key configuration concept is the use of make files which perform operations that extract explicit or tagged versions of specified files from revision controlled directories and produce the appropriate Software Sub-System., i.e. executable version of the software or software design documentation. In addition, make files will use only explicit rules for specifying versions of software that make up the deliverable. Therefore, by executing the baseline makefile, an executable version of software is produced that is traceable to it's composite source files. In a similar manner, by executing a document make file, design documents will be produced that describe the Software.

The ECO development process for software is similar to that for hardware except for the ECOs releasing software baselines. For all Software Sub-System baseline releases, the ECO process couples the design documents with the software executable file and release ID by identifying the name and revision control versions of the Baseline Make file and the Document Make file. In addition, the ECO identifies the previous version of the make files that the current version is based on. This enables audit-trail capabilities. Until approved, software will not be permanently installed in ACIS hardware.

5.11 Release Procedures

The need for control increases proportionally to the number of individuals that use the products of software development. As a result, different control procedures will be used depending on use.

- Preliminary software releases are identified with numeric revision identification and will be used early in the development process. These release identifiers will be used on software deliverables when sufficient functionality has been developed and can be used by others outside of the immediate software development team. When used in the above instance, the declared revision or release code i.e. 10, 11, 12, will be identified on the ECO and, if applicable, will appear in the executable code. Only the designer is required sign the ECO to signify that the statements on the ECO are correct.
- Baseline software releases are identified with alphabetic revision identification will be used later in the development process at specified milestones. For the SIS, these milestones are identified in the Software Development Schedule.

5.12 Change Control

Change control for software will begin during the integration phase and must start when software identified with a numeric release is given to someone outside of software development for use in their work. Change procedures as described in this document and the CMP are used throughout the life of the ACIS project.

- While a Preliminary Software Sub-System is used by individuals outside of software development, the following procedures will be used. The Software Sub-System release will be accompanied by an ECO that identifies the version of the release and briefly describes it's functionality.
- When a baseline release of a Software Sub-System is identified with alphabetic revision, all changes will be controlled as specified in the CMP. The installation of baseline software can occur only after the appropriate ECO has been approved by the SPM.
- When a baseline release is identified as a Flight Release and involves permanently loading the software into flight hardware, the PM will authorize the loading by ECO.

5.13 Problem Reporting

Problem reporting is used when software is released with numeric or alphabetic revision and is described below: In addition, Sofware Problem Reports (SPRs) will be identified in ECOs that resolve the issues brought up by the SPRs.

- When a release is identified with numeric revision, and is used by individuals outside of software development, a written means of problem identification is required to help ensure that the problems are attributed to the appropriate release. Anytime a problem is found and is attributed to software, a SPR will be used. Refer to Appendix 3. These SPRs will be given to the software developer for evaluation and a copy will be kept by SQA until the Software Sub-System ROM image file is released with an alphabetic revision ID.

- When a baseline release is identified with an alphabetic revision and is used during and after the Alpha release, all problems attributed to software will be recorded on SPRs. These SPRs will be evaluated and tracked by the SPM with assistance from SQA until no further action is required and the SPR is marked closed. A master report file will be maintained by SQA which contains supplementary data such as failure analysis and records of meetings.

5.14 Software Testing

A Software Test Plan (STP) will be written to satisfy the requirements found in the Detailed Description of the Software Development Process for Software Requirements Phase describing the Software Test Requirements Document section of the MM-8075.1. The STP may contain separate sections for each Software Sub-Section (SSS) The plan will provide management and the testing function with an overview of the test activities, schedules and resources required to perform testing. The plan will describe how the testing specifications found in the following sections will be implemented.

5.14.1 Unit Test;

All code will be unit tested to ensure that the individual unit (class) performs the required functions and outputs the proper results and data. Proper results are determined by using the design limits of the calling (client) function as specified in the design specification defining the called (server) function. Unit testing is typically white box testing and may require the use of software stubs and symbolic debuggers. This testing helps ensure proper operation of a module because tests are generated with knowledge of the internal workings of the module. Unit testing requirements and documentation guidelines are shown in Attachment 2.

5.14.2 Integration Test

There are two levels of integration testing. One level is the process of testing a software capability e.g. being able to send a message via a DMA port. or the ability to acquire a row of CCD data. During this level, each module is treated as a black box, while conflicts between functions or classes and between software and appropriate hardware are resolved. Integration testing requirements are shown in Attachment 2. Test cases must provide unexpected parameter values when design documentation does not explicitly specify calling requirements for client functions. A second level of integration testing occurs when sufficient modules have been integrated to demonstrate a scenario e.g. type of science mode or the ability to queue and receive commands. During this phase, composite builds, or baselines, of the software are married to the engineering versions of the hardware to evaluate the combined hardware/software performance for each operational function. The instrumentation and test equipment recommended in STP may be required to identify and resolve hardware /software deficiencies. Both hardware and software documentation is reworked a s necessary.

5.14.3 System Testing

System testing begins when sufficient hardware and software has been integrated that enable the operation and functions of the integrated ACIS Instrument. The purpose of system (stress) testing is to identify the operational envelope of the instrument. This is the responsibility of the Engineering Branch with assistance from the Science Branch. The Quality Branch will ensure that the stress tests have been run and the results recorded as specified in the STP. The SPM will ensure that the results of the stress tests are addressed where the results show that the operating envelope does not meet requirements identified in the Software Test Requirements Document (STRD - DM17).
Completion of system testing is dependent on availability of hardware. For the SIS, system test phase is complete just prior to the Flight release or until a hardware platform is no longer available for stress testing.

5.14.4 Validation Testing

Validation testing begins when sufficient hardware and software has been integrated that enables validation of the requirements identified in the Software Validation Test Specification (SWVATS - DM21) and System Acceptance Test Specification (SSATS - DM22). The purpose of validation testing is to ensure that the hardware/software system meets the science/interface requirements allocated to software as identified by the requirements Traceability method. For the SIS, SQA it is the responsibility of the V & V Function of the Quality Branch with assistance from the Engineering Branch to develop and run these tests. The PM will ensure that there will be sufficient physical and human engineering resources to support this V & V effort. Completion of validation testing is dependent on availability of hardware. For the SIS, validation test phase is complete just prior to the Flight release or until a hardware platform is no longer available.

Attachment 1 Coding Documentation Requirements

- A high level language shall be used except when approved by SPM.
- Each method, function and class will be identified with their own comment header. The contents of the header should identify the purpose and any assumptions the user or caller must be aware of.
- Coding documentation will, at a minimum, describe reasons for code branching and a description of each variable name at their point of memory allocation.
- Naming conventions shall be used that clearly distinguish literal constants, variables, methods and class/object names. Class/object names should be nouns,. methods should be verbs. variables shall not be re-used for different purposes, except in trivial cases such as loop counts and indices. In addition, all names will contain at least 2 (two) characters to facilitate global pattern searches.
- Coding complexity conventions for a Class shall be established, such as the use of the Cyclomatic Complexity Matrix. A description of how to calculate Cyclomatic complexity index can be found in Chap 13 of Software Engineering a Practitioners Approach by Roger S. Pressman.; McGraw-Hill. The design will not exceed a complexity index value (Vg) of 10, without the approval of the SPM.
- Dispatcher logic shall include a default clause, and loops shall include an escape clause except in `forever` loops.

Attachment 2 Testing Requirements

a. Unit Testing:

1. Environment; Specify testing environment. i.e. if and when stubs and drivers and/or other application routines, special hardware and/or conditions are to be used.
2. Logic Complexity: Calculate the Cyclomatic complexity matrix index which specifies the number of test cases required to ensure that all code is executed at least once. A description of how to calculate Cyclomatic complexity index can be found in Chap 13 of *Software Engineering a Practitioners Approach by Roger S. Pressman*, McGraw-Hill.
3. Boundary Analysis: Specify tests that will execute code using boundaries at *n-1, n, n+1*. This includes looping instructions, while, for and tests that use `LT, GT, LE, GE` operators.
4. Error handling: Design tests that verify the recording of all detected and reportable errors that a program is designed to find and report.
5. Global parameter modification: When a program modifies global variables, design tests that verify the modification. That is; initialize the variable independent of the program, verify memory contents, run the program, check that memory contents have been modified.
6. Mathematical limit checking: Design tests that use out of range values that could cause the mathematical function to calculate erroneous results.
7. Cessation of test: Specify the conditions under which a testing session stops and a new build is made. Regression testing is required, according to steps 2 through 6 above, of all lines of code that have been modified.
8. Documentation: The documentation must show that the tests have shown that the topics in items 2 through 6 above have been addressed.

b. Integration Testing;

This type of testing addresses the issues associated with the dual problems of verification and program construction. Integration is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design. The following topics are addressed in the STP.

1. Critical module definition: Decide which classes/modules contain critical control operations. These class/modules should be unit tested as soon as possible and not wait for subordinate class/object completion. Use of program stubs may be necessary.
2. Object grouping: Decide what modules comprise an integration group by use of scenarios and appropriate architecture diagrams. It is desirable to integrate at low levels to make bug definition easier. Choose objects that are related to a specific function like command uplink.
3. Depth vs. breadth testing: Decide how to test a group of objects/classes It is suggested that breadth testing be used when interfacing with the hardware. Use stubs, if required, to test dispatcher control modules. Use depth testing when a function is well defined and can be demonstrated, e.g. and application mode like timed exposure.

4. Regression testing: Integration regression testing is required whenever an interface attribute has been changed, e.g. the value of a passed parameter.
5. Top down vs. bottom up: Use top down testing to verify major control or decision points. Use bottom up to test hardware driver type programs.

c. System testing:

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Each test may have a different purpose, but all work to expose system limitations. System testing will follow formal test procedures based on hardware, software and science requirements as specified in the STP.

d. Validation Testing

The purpose of validation is to prove that the ACIS instrument performs as specified in the requirements documents listed above in the Applicable Documents Section of this document.

1. Validation tests/procedures will identify a testing method and pass/fail criteria.
2. When ranges are specified in the requirements, tests cases will include boundary values at *n-1, n, n+1* where possible.
3. When `LT` or `GT` limits are specified, the measured value should be recorded.

Testing Documentation:

Testing documentation must be sufficient to provide evidence that the testing objectives as stated in the preceding sections or the STP have been met.