

**Earth Observing System
Geoscience Laser Altimeter System**

**GLAS I-SIPS Software Configuration
Management Plan**

Version 1.0

May, 1999

Contact:

**R E. Jaquith, rjaquith@icesat2.gsfc.nasa.gov
Anita Brenner
David Hancock
Gladstone Marcus**

Table Of Contents

Table Of Contents	2
Document Change History	3
Reference Documents	3
Overview.....	4
Acronyms.....	5
Scope.....	7
Personnel Requirements.....	8
Configuration Manager	8
Software Engineering Coordinator and Quality Assurance Manager	8
Systems Administration/VOB Administration	8
Documentation Coordinator.....	8
Systems Test Manager	8
Process Control.....	9
Policies:.....	9
Process:.....	10
Change Control Board.....	12
Configuration Status Accounting	12
Software Development Library Authority	12
Change, Problem, and Version Control And Resolution.....	13
Build Process Control.....	16
Formal Build Process Flow.....	17
GLAS Distributed VOB Structure Example.....	18
Action Item Data Base	20
GNATS.....	20
Appendix 1: Initial GLAS ClearCase Configuration Specifications.....	21
V0_DEVELOP.....	21
gsfc_v0.....	22
gsfc_v0+V0_DEVELOP.....	22
gsfc_v0+WFF.....	22
gsfc_v1.....	22
gsfc_v1+V0_DEVELOP.....	23
gsfc_v1+WFF.....	23
wff+gsfc_v0	23
wff+gsfc_int.....	23
wff+gsfc_v1	23
PR_Sample (V1).....	23
Appendix 2: ClearCase.....	24
Attachments	Error! Bookmark not defined.
Ice Altimetry Problem Reports Submission.....	Error! Bookmark not defined.
GNATS Help	25
VOB Directory Structure For GLAS.....	27
Sample CM Software Checklist	28

Document Change History

Document Name: GLAS Software Configuration Management Plan		
Version Number	Date	Nature Of Change
1.0	May 25th, 1999	Original Document

Reference Documents

- GLAS Science Data Management Plan
- GLAS Science Software Management Plan
- GLAS Science Computing Facility Plan
- ICESat (GLAS) Science Processing Software Preliminary Design Review 11/24/98
- GLAS Computer Program Categories and Configuration Item Specifications
- ClearCase Fundamentals - P/N 4000-520-3b
- Action Item Data Base Users Guide - **TBD**
- GNATS Users Guide - **TBD**
- ICESat Internal Website Users guide - **TBD**
- GLAS SDT FORTRAN 90 Programming Standards And Guidelines
- "IEEE Standard Glossary of Software Engineering Terminology," IEEE-STD-610, ANSI/IEEE Std 610.12-1990, February 1991.
- "IEEE Standard for Software Configuration Management Plans," IEEE-STD-828, ANSI/IEEE Std 828-1983, June 1983.
- "Configuration Management," MIL-STD-973, Military Standard 973, April 1992.
- "Configuration Management Plan, Data Item Description," NASA-DID-600, NASA-STD-2100-9,1 NASA Software Documentation Standard, Software Engineering Program, July 1992.

Overview

This document is intended primarily to address the implementation of Software Configuration Management for the ICESat GLAS project. The focus is on the SCM elements relevant to the GSFC central SCF and the remote Wallops Flight Facility.

This document was prepared by Raytheon/ITSS personnel in support of B. E. Schutz, GLAS Science Team Lead for the GLAS investigation, via GLAS tasks 158 and 170, on the NASA/GSFC Geodynamics contract. The work was performed under the direction of David W. Hancock, III (Code 972) and Anita Brenner, the Raytheon/ITSS Department Manager.

David Hancock may be reached by phone at (757) 824-1238, by Electronic Mail at hancock@osb.wff.nasa.gov and by FAX at (757) 824-1036.

Anita Brenner may be contacted by phone at 301-614-5914, by Electronic Mail at Anita.Brenner@gsfc.nasa.gov , or by FAX at 301-614-5644.

Acronyms

AIDB	Action Item Data Base
ALT	EOS-ALTimeter spacecraft series
ANC	ANCillary
ATBD	Algorithm Theoretical Basis Document
CATU	The Central Automated Tracking Utility - Problem tracking utility, currently GNATS. For Each problem report (PR) , the person opening the PR must supply a CPC and a CI.
CC	Configuration Control
CCB	Change Control Board. The entity tasked with oversight and supervision of the program configuration elements (Hardware and Software)
CDDIS	Crustal Dynamics Data and Information System
CM	Configuration Management
CMDL	Climate Monitoring and Diagnostics Laboratory
CI	Configuration Item - The lowest level component in a CPC that is tracked in a configured environment. A CI equates to unit-level.
CPC	Computer Program Category - A method of categorizing computer related major elements. A CI is a component of a CPC. Each unit must have a CPC and a CI. In the GLAS environment.
CRF	Celestial Reference Frame
CSR	Center for Space Research at the University of Texas
DAAC	Distributed Active Archive Center
ECS	EOSDIS Core System
EDOS	EOS Data and Operations System
EOC	EOS Operating Center
EOS	Earth Observing System
EOSDIS	Earth Observing System Data and Information System
ESDIS	Earth Science Data and Information System
GDS	GLAS Ground Data System
GLAS	Geoscience Laser Altimeter System
GNATS	GN A Tracking System - A web-based defect tracking system from GNU
GPS	Global Positioning System
GSFC	NASA Goddard Space Flight Center at Greenbelt, Maryland
GSFC/WFF	NASA Goddard Space Flight Center/Wallops Flight Facility at Wallops Island, Virginia USA
HDF	Hierarchical Data Format
ID	IDentification
IEEE	Institute for Electronics and Electrical Engineering
IERS	International Earth Rotation Service
IGS	International GPS Service for Geodynamics
I-SIPS	ICESat Science Investigating and Processing System
IST	GLAS Instrument Support Terminal
ITRF	IERS Terrestrial Reference Frame
LASER	Light Amplification by Stimulated Emission of Radiation
LIDAR	Light Detection And Ranging
N/A	Not (/) Applicable
n/a	Not (/) Available
NASA	National Aeronautics and Space Administration
NOAA	National Oceanic and Atmospheric Administration
NSIDC	National Snow and Ice Data Center (DAAC)
PAD	Precision Attitude Determination
PI	Program Improvement
POD	Precision Orbit Determination
PR	Problem Report
QA	Quality Assurance
RINEX	Receiver INdependent Exchange
SA	Systems Administration
SCF	GLAS investigation Science Computing Facility and workstation(s)
SCM	Software Configuration Management

DRAFT

SDPS Science **D**ata **P**rocessing **S**egment
SPSO EOS **S**cience **P**roject **S**upport **O**ffice
STM **S**ystem **T**est **M**anager
TBD **T**o **B**e **D**etermined, **T**o **B**e **D**one, or **T**o **B**e **D**eveloped - Future/Deferred Activity
UNIX **U**niversal **I**nteractive **E**Xecutive - The operating system jointly developed by the AT&T Bell Laboratories and the University of California-Berkeley System Division
UTCSR **U**niversity of **T**exas **C**enter for **S**cientific **R**esearch
VOB **V**ersioned **O**bject **B**ase - An element of the ClearCase SCM Tool

Scope

The GSFC central SCF has been designated as the GLAS primary development and production site for Investigator-led Science Processing Software (I-SIPS). The ultimate responsibility for source and ancillary components (e.g. Executables and documentation) resides in the I-SIPS. The plan also recognizes the criticality of the development efforts at the WFF. The development of the GLAS standard software at GSFC and the WFF, will be implemented and maintained using Rational Software's ClearCase SCM software. Additional software and data inputs will be delivered to the GSFC facility from the University of Texas Center for Scientific Research (UTCSR). The UTCSR software generates the Precision Attitude Data (PAD) and Precision Orbit Data (POD), used for GLAS production. The UTCSR PAD software and data will be primarily under the control of that facility and will be delivered to GSFC as versioned software packages, which include the source code and executables. The UTCSR versioned PAD software packages will then be placed under the control of ClearCase at the GSFC. UTCSR will be responsible for versioning the POD software. The GLAS development effort will be distributed, requiring close coordination between facilities, and will employ multiple automated tools and processes to be successful.

Personnel Requirements

Configuration Manager

The Configuration Manager is responsible for the overall planning and implementation of the SCM for the GLAS I-SIPS. This individual coordinates the CM policy with the Software Lead. The CM works closely with the Quality Assurance Manager to ensure compliance with customer requirements and project policies. This person is also the primary individual who produces the certified program and formal integration builds for delivery to the System Test Manager. This individual supports the test manager in integration and acceptance testing

Software Engineering Coordinator and Quality Assurance Manager

This person is tasked with overseeing the software engineering processes. He coordinates the day-to-day activities of remote and the primary site and overlooks the design processes. Close coordination with CM is imperative. The Software Engineering Coordinator also acts as the Quality Assurance (QA) Manager, and ensures that all applicable policies and procedures are adhered to in the creation of documentation, software and products in the I-SIPS environment at all SCF sites. QA is responsible for scheduling and performing process and product audits and for reporting discrepancies to the software development team and the CCB.

Systems Administration/VOB Administration

As there are several distributed systems employed in the ICESat/GLAS configuration, SA tasks are necessarily distributed amongst several persons. The lead SA Manager at the primary site will coordinate the SA tasks. SA adherence to and support of CM/QA/Software Engineering is *crucial* to the success of the project. SA can ensure proper implementation of the SCM plan. SA tasks include creating and maintaining the ClearCase VOB structures, implementing a multi-layered authorization schema, and performing VOB administration functions.

Documentation Coordinator

This person is tasked with the maintenance, cataloging and approval process for ALL documentation used in the project. This includes documentation provided with COTS and project developed documentation. There must be close coordination with the CM, SEC and QA managers. This person may also be thought of as a data library coordinator.

Subsystem Lead

This person is tasked to oversee the design and development of one or more subsystems of the GLAS I-SIPS software. The subsystem lead will arrange reviews and create documentation for the design, coding and unit testing of the subsystem elements. This person will coordinate activities with the CM, QA and Test managers to maintain proper configuration control of the software.

Systems Test Manager

This person is tasked with preparing and implementing the integration and acceptance test procedures, as well as reviewing unit test results. The System Test Manager also directs test data preparation and all end-to-end simulation processes. The STM will closely coordinate with CM, QA and the Subsystem Leads to ensure complete and rigorous testing of all software and hardware components before their inclusion into the certified libraries and the system hardware configuration. The STM will submit the results of testing to the Software Engineering Coordinator, CM, and to the Subsystem Lead.

Process Control

There will be 4 categories of processes that will be controlled as part of GLAS SCM. These are integration test failures, version problems, design changes, and version releases. Each of these categories will be tracked using a Central Automated Tracking Utility (CATU) which will be implemented using GNATS.

Policies:

- After design documents for the I-SIPS software have been approved and “locked down”, developers are expected to adhere to the design and to follow a change procedure when a deviation from the design is requested. The developer will initiate the **design change** by using the CATU and opening a CR via the user interactive interface.
- A CR will not be required if during *implementation it is decided to decompose a module on the structure chart but the composite of the resultant modules contains the same input and output as the original.*
- If the CCB approves the CR then the CM will assign the CR to the appropriate subsystem lead. The Subsystem Lead may then assign the CR to the appropriate individual(s), as required.
- After a version has been released, and a problem found, then a Problem Report (PR) is opened using the CATU. The CATU automatically notifies the CM of any PRs and the CM will prepare it for presentation to the CCB.
- If the CCB approves the PR then it is assigned to a subsystem lead who may assign it to the appropriate individual.
- PR and CR numbers will be defined as PRmmddyy or CRmmddyy followed by the 3-digit sequence identifier, nnn. The sequence identifier denotes that this was the nth such request of this type created on this calendar date.
- PDL (pseudocode) will be checked in as the 1st version on the **integration** branch..
- The software developers will perform their work on a development branch, which is a branch derived from the **integration** branch, for each element.
- The Subsystem Leads are authorized to merge **updated** software onto the **integration** branch after review and approval by an assigned primary reviewer. Labeling of the elements is done at this time.(e.g. CREV_PASS)
- Any problems that occur during integration testing will be tracked using an integration problem report, IPR. The test manager initiates the IPR in CATU which automatically notifies the Software Engineering Coordinator. No action by the CCB is required. The Software Engineering Coordinator is responsible for tracking IPRs.
- ONLY the CCB may authorize the merge of any element onto the element **main** branch.
- Only the CCB may authorize the release of a version.
- The CCB will initiate and track version releases by opening a version request, VR, in the CATU.
- All formal releases are released from the main branch.
- ONLY the Configuration Manager may perform the actual merge from the **integration** branch to the element **main** branch at the direction of the CCB.
- Only the System Test Manager may certify that a build has passed testing on the integration and main branches.
- Developers will apply the **CODE_REVIEW** meta-data label to checked-in element(s) on the development branch when the element(s) in question are ready for peer/Subsystem Lead review.
- Subsystem Leads will apply the **CREV_FAIL** or **CREV_PASS** meta-data labels to element(s).
- Subsystem Leads are responsible for performing the merge of the element(s) to the **integration** branch. The **BUILD** meta-data label is automatically applied to the element(s), at this time to indicate that the element is eligible for integration build testing.
- ONLY the System Configuration Manager may apply the **BUILD_PASS** or **BUILD_FAIL** meta-data labels to elements that are **integration build** tested.
- ONLY the System Test Manager may apply the **INT_PASS**, **INT_FAIL** meta-data labels to elements on the **integration** branch.
- The CCB will authorize the CM to apply **VERSIONn.n** labels to elements on the integration branch that have passed integration testing
- The CCB will authorize the CM to apply **RELEASE_n.n** labels to versions of an element on the main branch
- The CCB will authorize the CM to apply **RELEASED** label to versions of an element on the main branch to indicate a version that was included in the production build.

Process:

The following list presents the typical development/element change processes. It outlines the steps to be taken and the order in which they are normally accomplished. *Steps unique to the creation of a new element are italicized.*

- *For new elements*
 1. *Program Development Language (PDL) for the element will be created by the software development team and reviewed by the Software Engineering Coordinator and peer reviewer (s).*
 2. *The developer will use the ClearCase **mkelem** command. He/she will use a config spec (see sample 1 of appendix A) set up to automatically create this element on the main branch, branch to the integration branch, and branch to a development branch.*
- For changes and problems
 1. A Change Request (CR and PR) will be submitted using CATU. (GNATS will automatically assign it a number.)
 2. The originator will fill in the appropriate data fields on the form electronically, using CATU. The CM will complete the form and present it to the CCB at the next regularly scheduled meeting.
 3. If the CCB approves the CR or PR, they will assign it to a Subsystem Lead.
 4. The VOB administrator will create a branch type for the CR or PR with the identification number. An example of this branch name would be **PR052599001**.
 5. The developer will modify his or her config spec file to use this new branch type as the development branch for all elements involved in satisfying this request.
- The element will be modified as required by the assigned developer and when that developer has finished coding modifications he/she will assign the meta-data label **CODE_REVIEW**. The **CODE_REVIEW** label indicates only that the element is ready for peer and Subsystem Lead code review.
 - For elements that pass this review process, The Subsystem Lead will apply the meta-data label **CREV_PASS** indicating the element is ready for unit testing.
 - For elements that fail the review process, the Subsystem Lead will apply the meta-data label **CREV_FAIL** and notify the assigned developer.
- The developer then prepares a set of unit tests, which must be reviewed by the assigned reviewer. After the developer is satisfied that he/she has completed unit testing the module, he/she will apply the label **UTEST_REV**
- The assigned reviewer will then review the unit test results
 - If the unit tests are successful:
 - The subsystem Lead will be notified and will apply the label **UTEST_PASS**
 - If the unit tests are unsuccessful:
 - The subsystem Lead will be notified and will apply the label **UTEST_FAIL**
- After the **UTEST_PASS** label has been applied to an element, the subsystem lead will merge it onto the integration branch where the **BUILD** label is automatically applied, an entry will be automatically entered in a log file which will be mailed to the CM on a daily basis.
- The Configuration Manager will regularly attempt the **formal integration build** process, which will include all eligible elements, that is, those with the **BUILD_PASS** label.
- If the formal integration build fails, the Configuration Manager will isolate the failing elements and will:
 - Notify the Subsystem Lead(s) responsible for the failing element(s)
 - Update the Problem Report (PR) if one exists
 - Notify the CCB if this is in response to a problem report and also the Software Engineering Coordinator
 - Apply the meta-data label **BUILD_FAIL** to the checked in element(s) and submit a problem report to the SEC, using CATU.
- If the formal integration build succeeds:
 - The Configuration Manager applies the **BUILD_PASS** meta-data label to the checked-in element(s).
- The Configuration Manager then notifies The System Test Manager and updates the PR.
- The System Test Manager performs specified integration tests on the newly created **formal integration build** and generates an integration test report.

DRAFT

- If the integration test fails , the System Test Manager will:
 - Isolate the offending element(s) and notify the Configuration Manager who will then re-generate the build without compiler optimization to aid in diagnosis and to verify that compiler optimization did not cause the build failure.
 - The System Test Manager will then re-run the previous tests.
 - If the build test still fails, a request will be sent to the Configuration Manager to generate a new build with debugging mode in effect.
 - The previous tests will be re-run and the outputs collected for detailed analysis. The appropriate Subsystem Leads will participate in the analysis of the failure(s).
 - The System Test Manager will apply the meta-data label **INT_FAIL** to the offending element(s).
 - The PR for the offending element(s) will be updated if it exists
 - Notify the CCB if this is in response to the PR and always the Software Engineering Coordinator
- If the integration test succeeds:
 - The System Test Manager will apply the meta-data label **INT_PASS** to all elements used in this build and will also apply the version label **VERSIONn.n** to the element(s).
 - The System Test Manager will update the PR for the new element(s) and will change the PR status to **tested**, signifying that the PR(s) in question have passed formal testing.
- The System Test Manager will inform CM and the CCB of the test outcome.
- The CCB will evaluate the results:
 - For failed tests the CCB will decide the course of action. They can decide to close the PR with an explanation and go no further.
 - For successfully tested PRs the CCB will either authorize CM to merge the new software element(s) on to the main branch(es) with both the meta-data labels '**RELEASEn.n**' and '**RELEASED**', or will specify that the element(s) be merged on to the main branch with only the '**RELEASEn.n**' meta-data label applied.¹
- The Configuration Manager will perform a formal **production** build.
- If the **production build** succeeds:
 - The Configuration Manager will apply the **PBUILD_PASS** label to the element(s) **main** branch and notify The System Test Manager and the CCB.
 - The System Test Manager will perform Acceptance testing of the newly created **production build** to ensure that it is suitable for release.
 - If the **Acceptance test** is successful:
 - The System Test Manager will apply the **PTEST_PASS** label to the element(s) **main** branch and notify the CCB which will determine if the
 - If the **Acceptance test** fails:
 - The System Test Manager will apply the **PTEST_FAIL** label to the element(s) **main** branch and notify the CCB
- If the **production build** fails:
 - The Configuration Manager will apply the **PBUILD_FAIL** label to the element(s) **main** branch and notify the CCB.

¹ Note that if the RELEASED label is not present, the element will not be included in the formal **production** build.

Change Control Board

The Change Control Board has the responsibility for tracking and verifying changes to the overall program resources, whether programmatic or technical. This board receives, evaluates and recommends approval or rejection of the proposed change. The CCB assigns that change request to the appropriate individual.

Weekly reports incorporating statistical data on the following areas are required from the Subsystem Leads to the board:

- The number of Problem Reports **opened** for the week.
- The number of Change Requests **opened** for the week.
- The number of Problem Reports **rejected** for the week.
- The number of Change Reports **rejected** for the week.
- The number of Problem/Defect Reports **updated** for the week.
- The number of Change Requests **updated** for the week.
- The number of Problem/Defect Reports listed as **closed** for the week.
- The number of Change Requests **closed** for the week.
- The number and type of problems and defects which **failed** in testing.
- The number of Change Requests which **failed** in testing.
- The number and type of problems and defects which were **tested**.
- The number of Change Requests which were **tested**.

The CCB will track the program to the configuration item level (a CI), but not necessarily to the individual subroutine(s). In essence, a subroutine will be included in a Computer Program Category (CPC) or in a CI, but the subroutine may, or may not, be a CI itself

The CCB will be composed of personnel from the following disciplines:

- Software Development Management
- Quality Assurance
- Test Management
- Operations Management
- Configuration Management
- Science Team Representative

The responsibility for ensuring compliance with the configuration control policy rests with the Software Development Management, and the Software Engineering Coordinator who will direct the individual Subsystem Leads.

Configuration Status Accounting

The purpose of configuration status accounting is to maintain the version status records for all Configuration Items (CIs) and make the current status records available to management in the form of status accounting reports. These reports include a list of the CIs, their current version numbers, and the status of all non-closed PRs and CRs against those CIs.

Status accounting reports will be generated by the CM and distributed to management on a periodic basis. Configuration status accounting will be performed regularly, at least monthly to ensure program level oversight..

The CPCs and CIs are maintained in a separate reference document, GLAS Computer Program Categories and Configuration Item Specifications.

Software Development Library Authority

DRAFT

The SDL contains all the software that was controlled through both developmental and formal configuration control (CC).

Software Developers for I-SIPS will have equivalent access authority at each of the respective sites. While developers may check out code for their assignments from either the **main branch** or an **integration** branch, only CM personnel may check code into the **main branch**. Developers are restricted from checking in on a developer created "modification" (problem/change request) branch.

The authorities assigned to individual development personnel for the GLAS development effort are as follows:

- Developers may check-in modules to their development/problem/change request branches.
- Subsystem Leads accept unit-tested elements from the developers and, after reviewing, check-in those elements to the site **integration** branch.
- CM personnel check-in the fully tested elements from the site integration branch to the element **main** branch² upon approval of the CCB.

Change, Problem, and Version Control And Resolution

The primary tool for tracking software problems in the ICESat/GLAS environment is the Central Automated Tracking Utility (CATU). All GLAS software developers and product users are expected to use the CATU when reporting a change request, problem request, integration problem report, or version request. The CATU will ensure that standardized reports will be generated for program management.

The following are labels applied in the problem resolution process:

- **Open** - This is a newly opened problem or defect.
- **Assigned** - This problem or defect has been assigned to the appropriate personnel for corrective action.
- **Updated** - This item has been marked as updated, meaning the corrective action has been completed by the assigned personnel and is now available for system testing, integration and verification.
- **Rejected** - This problem or defect report was determined to be incorrect or a duplicate of an existing problem, which is in processing or has been verified as already resolved.
- **Tested** - This element has passed testing and is eligible for CCB review.
- **Failed** - The corrective action for the PR failed to satisfy the requirements and the problem is re-assigned to the person tasked with the corrective action. A copy of the test report is sent to the CCB.
- **Closed** - This item has passed system test requirements and fully resolves the original problem, defect or action item. The CCB may, with an appropriate explanation close the PR or CR. In any case, closing a PR may only be done with the concurrence of the CCB.

² CM personnel may also check-in elements from any branch as required, but will normally only check-in elements as described.

DRAFT

The following will trigger the use of the CATU in problem resolution:

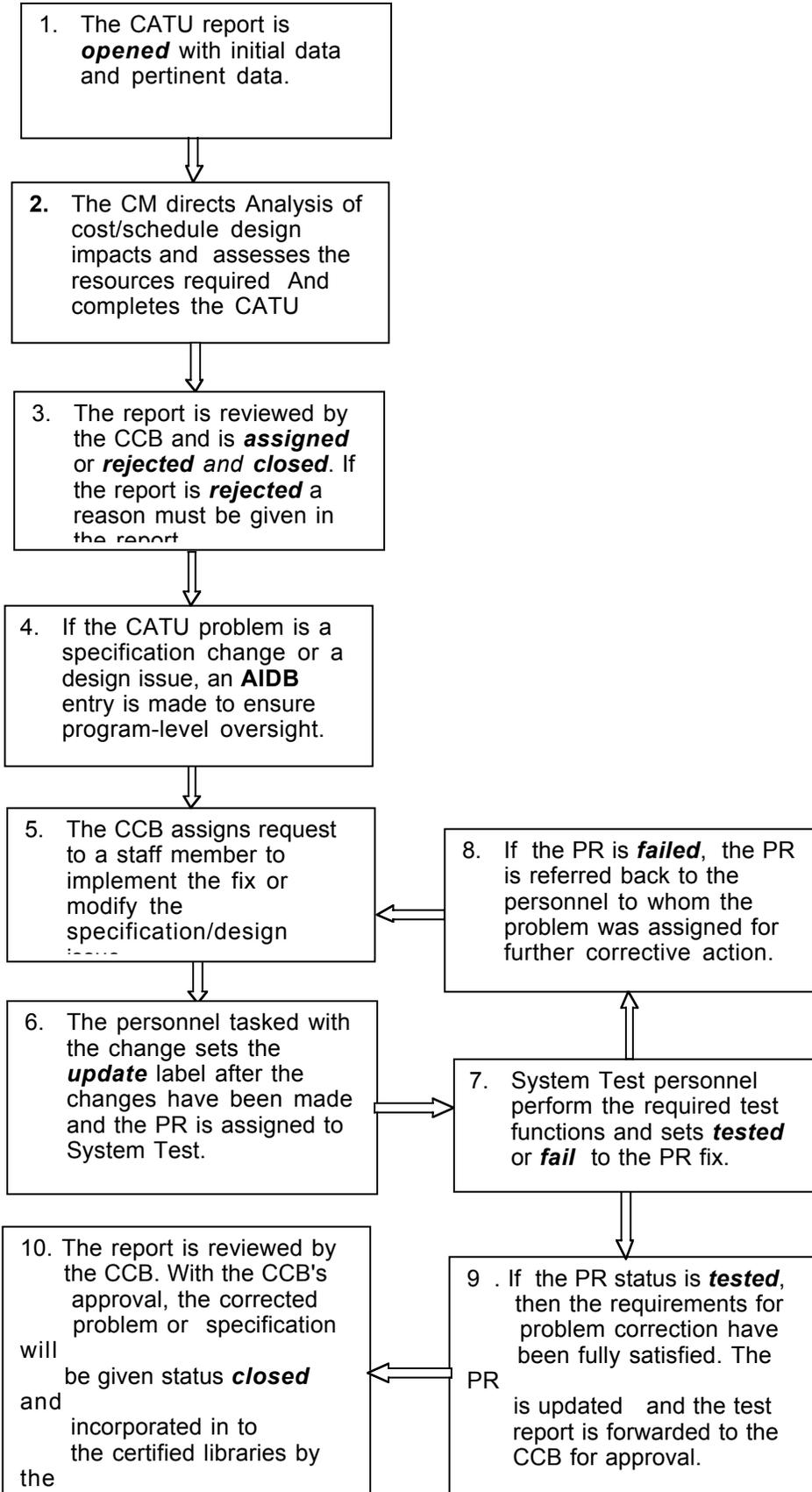
- (a) A software specification requirement
- (b) A software test outcome, at the integration test level
- (c) A design issue
- (d) A Software/Hardware interface problem
- (e) A build process failure
- (f) A configuration management tool problem, e.g. ClearCase or the CATU itself.
- (g) A System software problem, for example HP-UX or other COTS software failure in the development processes
- (h) A new version request

When a CATU report is initiated it ***MUST*** include the following at a minimum³:

- (a) A precise description of the problem or specification change or elements to include in the version.
- (b) A complete description of the software/hardware/process affected.
- (c) Any inputs required to duplicate the problem, or in the case of a specification or design issue, a description of the deficiency or new requirement.
- (d) The originator's name and task area.
- (e) The criticality of the problem or change specification.
- (f) The date for the required resolution
- (g) The initial person to whom the action is assigned.
- (h) The initial priority of the action
- (i) For problems,
 - (j) The host machine where the problem was detected.
 - (k) The time that the problem occurred and .
 - (l) The version or Build of the software in which the problem was detected.

³ Note: The priority, criticality, and assignment data may be revised by the CCB.

The following illustrates the CATU Flow:



Build Process Control

The build process is deeply intertwined with the test processes in that a **formal** build of the delivered software must be accomplished prior to the build package being made available to the test team. CM personnel will have reviewed the software package and will certify that the package conforms to project standards, but CM will not certify the efficacy of that package.

The build personnel will integrate the configured software into an integration test package, suitable for handoff to the designated test team for formal testing.

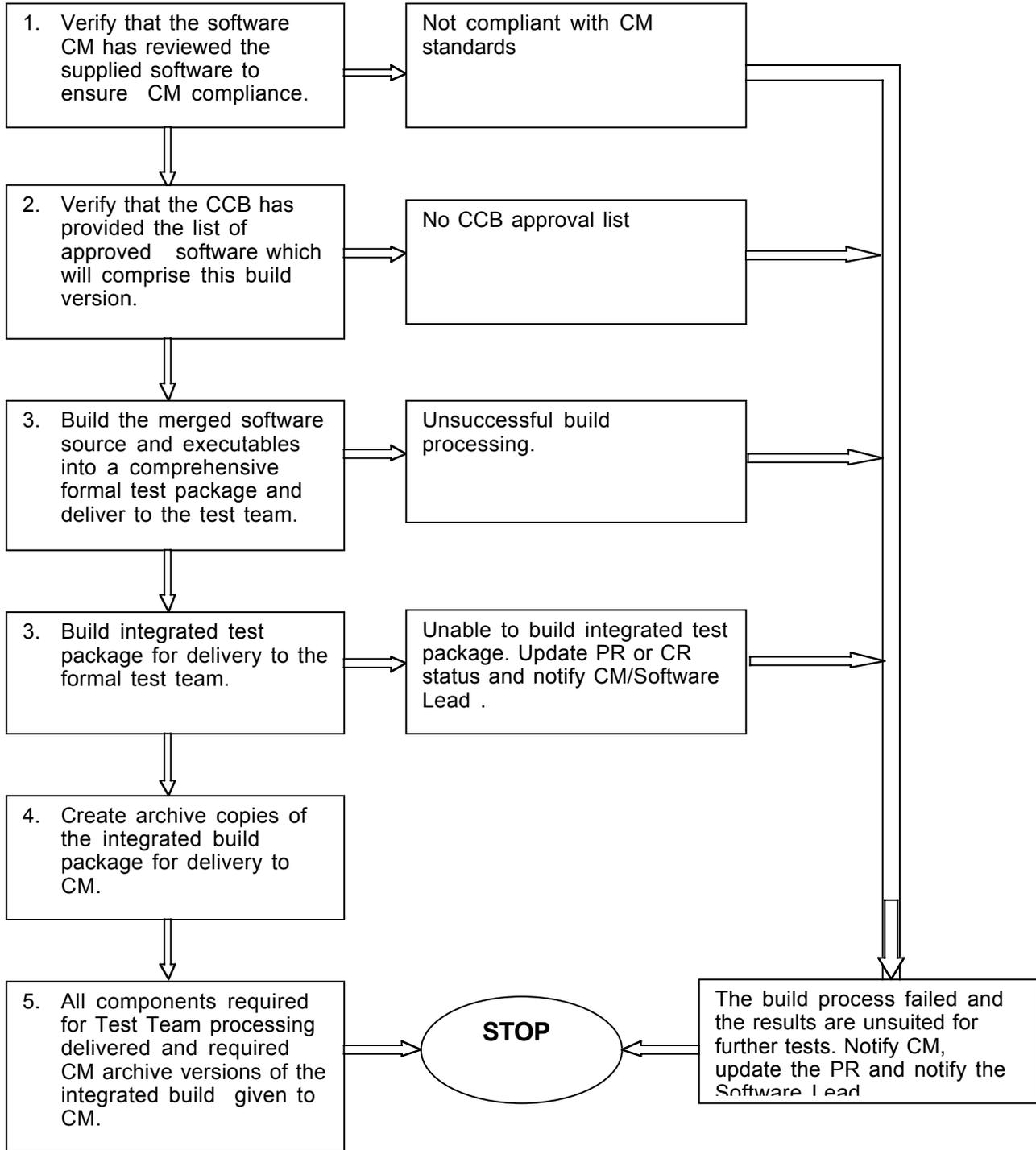
If the build process fails due to code deficiency:

- The test team will be informed and a Build report will be submitted to the SEC. A copy will be kept in the Unit Development folder in the "/build" directory. A PR will be created, or an existing one amended with relevant data and descriptions, and the integration request returned to the Subsystem Lead who authorized the package for build/test.
- If the code changes were initiated for an existing PR, then the PR status will be changed to **failed**.
- If the code changes were initiated for an existing CR, then the CR status will be changed to **failed** and an integration PR for the failing element(s) will be sent to the SEC.

The following list details the steps, which must occur during the build phase, prior to formal system testing:

1. The Configuration Manager must certify that the software and documentation required for the build process are complete and accessible. If all components are NOT in compliance, the Configuration Manager will amend the PR and return it to the Subsystem Lead with a list of the deficiencies for corrective action.
2. A build specification list will be created by the Configuration Manager to specify the CCB approved modules which are to be included in this build.
3. The Configuration Manager will create a complete system build using the approved software.
4. The Configuration Manager will version the newly created build for the test team and create 2 archive copies of the completed build, normally in TAR format. These archive copies will include, where practical, the requisite data inputs used for the subsequent tests.
5. Upon a successful build, implying a merge, compilation, SDT notification, and the creation of the requisite executables, the Configuration Manager will collect and identify all components necessary for the test team to commence its test functions and hand-off the build media to the Test Manager.

Formal Build Process Flow



DRAFT

GLAS Distributed VOB Structure Example

Action Item Data Base

TBD

GNATS

GNATS was designed as a tool for software maintainers. It consists of several utilities which, when used in concert, formulate and administer a database of Problem Reports grouped by site-defined "problem categories". It allows a support organization to keep track of problems (hence the term "Problem Report") in an organized fashion. Essentially, GNATS acts as an active archive for field-separated textual data submitted through electronic mail. Additional information is available from <http://www.gnu.org/software/GNATS/GNATS.html>

In the ICESat/GLAS environment GNATS is the problem control mechanism employed. All problems and requests for program changes MUST be initiated via the GNATS process. A sample CATU Problem Report (PR) is shown in the Appendix.

Appendix 1: Initial GLAS ClearCase Configuration Specifications

The following configuration specifications (config_specs) are documented internally for developer use on the I-SIPS GLAS project. These files have been tested and comprise the bulk of the required specification ***files but not all scenarios can or should be documented here.***

These Config_Specs are located in the `/VOBS/GLAS/cc_util/config_specs` directory and are available at both Goddard Space Flight Center and the Wallops Flight Facility. The following table describes the files and their purpose.

Configuration Specification File Name ⁴	Purpose
V0_DEVELOP	Config_spec for performing V0 development at Wallops
gsfc_v0	Config_spec for performing Version 0 development at Goddard Space Flight Center
gsfc_v0+V0_DEVELOP	Config spec for performing Version 0 development at Goddard while allowing view of Latest versions of elements on V0_DEVELOP branch IF AND ONLY IF there are NO versions on either the <code>gsfc_v0</code> or <code>gsfc_int</code> branches.
gsfc_v0+WFF	Config spec for performing Version 0 development at Goddard, that allows user to see latest version of an element on the WFF branch, IF AND ONLY IF there is NO version of the element on either the <code>gsfc_v0</code> or <code>gsfc_int</code> branches.
gsfc_v1	Config_spec for performing Version 1 development at Goddard Space Flight Center.
gsfc_v1+V0_DEVELOP	Config spec for performing Version 1 development at Goddard while allowing view of Latest versions of elements on V0_DEVELOP branch IF AND ONLY IF there are NO versions on either the <code>gsfc_v1</code> or <code>gsfc_int</code> branches.
gsfc_v1+WFF	Config spec for performing Version 1 development at Goddard, that allows user to see latest version of an element on the WFF branch, IF AND ONLY IF there is NO version of the element on either the <code>gsfc_v1</code> or <code>gsfc_int</code> branches.
wff+gsfc_v0	Config spec for performing Version 0 development at Wallops, while allowing developer to view latest version of an element on the <code>gsfc_v0</code> branch IF AND ONLY IF there are no versions of the element on either the WFF or V0_DEVELOP branches.
wff+gsfc_int	Config spec for performing Version 0 development at Wallops, while allowing developer to view latest version of an element on the <code>gsfc_int</code> branch IF AND ONLY IF there are no versions of the element on either the WFF or V0_DEVELOP branches.
wff+gsfc_v1	Config spec for performing Version 0 development at Wallops, while allowing developer to view latest version of an element on the <code>gsfc_v1</code> branch IF AND ONLY IF there are no versions of the element on either the WFF or V0_DEVELOP branches.
PR_Sample	This Config spec file is not intended to serve as an actual PR branch, only to illustrate and to serve as a template to be adapted into an existing Config Spec file.

V0_DEVELOP

```
Element * CHECKEDOUT
element * /main/WFF/V0_DEVELOP/LATEST
element -file * /main/WFF/LATEST -mkbranch V0_DEVELOP
```

⁴ These filenames are case-sensitive and are exactly as they appear in the config_specs directory.

```

element * /main/WFF/LATEST
element * /main/LATEST -mkbranch WFF

```

gsfc_v0

```

Element * CHECKEDOUT
element * /main/gsfc_int/gsfc_v0/LATEST
element -file * /main/gsfc_int/LATEST -mkbranch gsfc_v0
element * /main/gsfc_int/LATEST
element * /main/LATEST -mkbranch gsfc_int

```

gsfc_v0+V0_DEVELOP

```

Element * CHECKEDOUT
element * /main/gsfc_int/gsfc_v0/LATEST
element -file * /main/gsfc_int/LATEST -mkbranch gsfc_v0
element * /main/gsfc_int/LATEST
element * /main/WFF/V0_DEVELOP/LATEST -nocheckout
element * /main/LATEST -mkbranch gsfc_int

```

gsfc_v0+WFF

```

Element * CHECKEDOUT
element * /main/gsfc_int/gsfc_v0/LATEST
element -file * /main/gsfc_int/LATEST -mkbranch gsfc_v0
element * /main/gsfc_int/LATEST
element * /main/WFF/LATEST -nocheckout
element * /main/LATEST -mkbranch gsfc_int

```

gsfc_v1

```

Element * CHECKEDOUT
element * /main/gsfc_int/gsfc_v1/LATEST
element -file * /main/gsfc_int/LATEST -mkbranch gsfc_v1
element * /main/gsfc_int/LATEST
element * /main/LATEST -mkbranch gsfc_int

```

gsfc_v1+V0_DEVELOP

```

Element * CHECKEDOUT
element * /main/gsfc_int/gsfc_v1/LATEST
element -file * /main/gsfc_int/LATEST -mkbranch gsfc_v1
element * /main/gsfc_int/LATEST
element * /main/WFF/V0_DEVELOP/LATEST -nocheckout
element * /main/LATEST -mkbranch gsfc_int

```

gsfc_v1+WFF

```

Element * CHECKEDOUT
element * /main/gsfc_int/gsfc_v1/LATEST
element -file * /main/gsfc_int/LATEST -mkbranch gsfc_v1
element * /main/gsfc_int/LATEST
element * /main/WFF/LATEST -nocheckout
element * /main/LATEST -mkbranch gsfc_int

```

wff+gsfc_v0

```

Element * CHECKEDOUT
element * /main/WFF/V0_DEVELOP/LATEST
element -file * /main/WFF/LATEST -mkbranch V0_DEVELOP
element * /main/WFF/LATEST
element * /main/gsfc_int/gsfc_v0/LATEST -nocheckout
element * /main/LATEST -mkbranch WFF

```

wff+gsfc_int

```

Element * CHECKEDOUT
element * /main/WFF/V0_DEVELOP/LATEST
element -file * /main/WFF/LATEST -mkbranch V0_DEVELOP
element * /main/WFF/LATEST
element * /main/gsfc_int/LATEST -nocheckout
element * /main/LATEST -mkbranch WFF

```

wff+gsfc_v1

```

Element * CHECKEDOUT
element * /main/WFF/V0_DEVELOP/LATEST
element -file * /main/WFF/LATEST -mkbranch V0_DEVELOP
element * /main/WFF/LATEST
element * /main/gsfc_int/gsfc_v1/LATEST -nocheckout
element * /main/LATEST -mkbranch WFF

```

PR_Sample (V1)

```

Element * CHECKEDOUT
element * /main/gsfc_int/gsfc_v1/LATEST
element -file * /main/gsfc_int/gsfc_v1/LATEST -mkbranch PR042399001
element * /main/gsfc_int/LATEST
element * /main/LATEST -mkbranch gsfc_int

```

Appendix 2: ClearCase

Rational Software's ClearCase product provides a comprehensive configuration management solution, including version control, workspace management, build management, and process control. ClearCase offers a uniquely transparent, non-intrusive approach, and supports multiple platforms and IDEs, making it relatively easy to deploy and maintain.

- Goes beyond version control by also offering powerful workspace management, build management and process control capabilities.
- Enables parallel development across geographically distributed sites.
- Integrates with Microsoft® Developer Studio™, PowerBuilder®, Oracle Developer/2000™ and many of Rational's software development solutions.
- Provides disconnected usage model, so that users can work at home with ease and then reliably merge work back into the main line of development.
- Offers advanced build auditing; enabling teams to guarantee what went into any build.
- Performs automatic, graphical merges of files and directories highlighting code conflicts and safely resolving inconsistencies.
- Versions everything that evolves in development - including source code, binaries, executables, documentation, test suites, libraries, and directories.
- Tightly integrated with change request management products to track which defects were fixed in each release.

Additional Information on ClearCase can be obtained by referencing the ClearCase documentation, and at: <http://www.rational.com/products/clearcase>

GNATS Help

PR Number

This field contains the number of the PR to retrieve.

Originator

This field contains the full name of the person who originally submitted the PR.

Severity

This field is used to indicate your perception of the impact of the problem on the normal functioning for the IA system. In general, the Non-critical severity should be used for minor problems or suggestions (or compliments ;^)). Critical should only be used for crashes or other problems, which prevent the system from being used at all, or have an extreme impact on usability or performance. All other problems should be set to Serious . For queries, the severity is restricted to the listed choices.

Priority

This field is used to indicate your perception of the priority this problem should be assigned by the IA team. The priority of your PR may be reassigned by the IA team due to changing work priorities. For queries, the priority is restricted to the listed choices.

Category

This field is used to indicate your perception of the general category of this problem. The category of your PR may be reassigned by the IA team due to more detailed analysis of the problem. For queries, the category is restricted to the listed choices.

State

This field is used to indicate the state of the PR(s) you are querying for. PRs start life in the open state. When they are initially examined and analyzed, they move to the analyzed state. When the problem has been addressed and the fix is being tested, the PR moves to the feedback state. When all changes have been integrated, documented and tested, the PR is marked as closed . If work on a PR has been postponed, the PR is marked suspended .

Responsible

This field contains the name of the individual who has been assigned the responsibility of addressing a particular PR.

Simple Free Text Search

This field is used to specify a short text string for a quick search. This text string is searched for in the following fields of the PR: Submitter-Id, Category, Arrival-Date, Originator, Release, Synopsis, and Responsible . For advanced users, this field can accept grep -style regular expressions .

Detailed Free Text Search

This field is used to specify a short text string for a detailed search. This text string is searched for in the following fields of the PR: Organization, How-To-Repeat, Unformatted, Environment, Fix, Description, Audit-Trail . For advanced users, this field can accept grep -style regular expressions .

Report Format

Use this popup to specify the format of the report for your query results. The Header format only contains a subset of the PR fields. The Full format contains the complete text of the PR record. The Summary format gives a one-line summary of each PR, similar in content to the Header format. The SQL format is also a one-line format, suitable for use in uploading the information to a database.

Originator *Mandatory*

This field contains the full name of the person completing the PR. Non-IA System Team Members are required to provide their organization name, address and phone number where they can be reached during business hours. Originator Internet address is optional and recommended.

Synopsis *Mandatory* .

The PR title should be concise, yet descriptive of the problem or enhancement. The title is useful for scanning the PR database for a specific PR. When the title is ambiguous or nondescript, the capability to isolate a specific PR is greatly reduced.

Confidentiality

If this box is checked, your PR will be treated as confidential information. The primary effect of this box is to disallow querying of this PR via electronic mail or the Web.

Severity

This field is used to indicate your perception of the impact of the problem on the normal functioning of the IA system. In general, the Non-critical severity should be used for minor problems or suggestions (or compliments ;^)). Critical should only be used for crashes or other problems, which prevent the system from being used at all, or have an extreme impact on usability or performance. All other problems should be set to Serious.

Priority

This field is used to indicate your perception of the priority that should be assigned to this problem by the IA team. The priority of your PR may be reassigned by the IA team due to changing work priorities.

Category

This field is used to indicate your perception of the general category of this problem. The category of your PR may be reassigned by the IA team due to more detailed analysis of the problem.

Host Machine

Indicate the name of the machine the Client is running on (e.g., harp.gsfc.nasa.gov). If unknown, leave blank.

Problem Description/Proposed Enhancement *Mandatory* .

- A detailed description of software problem or proposed enhancement. Problem descriptions should include:
- Conditions at the time the problem was encountered (e.g., what window was active, what other windows were open, test script ID and step number).
- Type of machine (e.g., SGI, Sun, HP, IBM), operating system (e.g., Irix, SunOS), and operating system version number (Irix 5.2, SunOS 4.1.3)
- Problem symptoms, (what was expected to happen versus what actually happened).
- Date and time of problem occurrence.
- Proposed enhancement descriptions should be clear and concise. Include enough detail to support a detailed analysis.

VOB Directory Structure For GLAS

/GLAS_VOB

