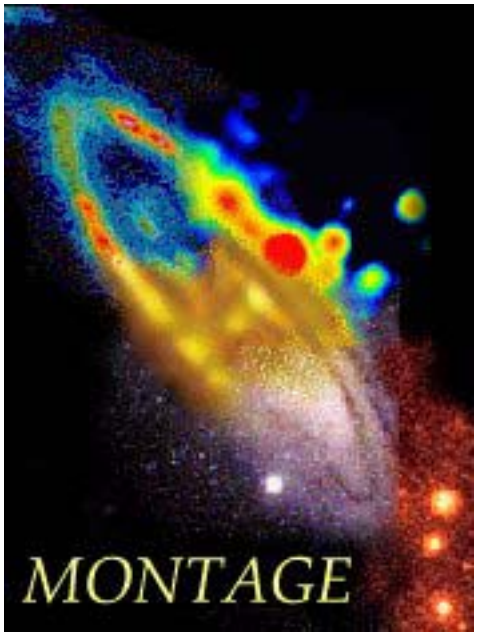

Software Test Plan for

MONTAGE

An Astronomical Image Mosaic Service for the National Virtual Observatory

Version 1.0 (September 24, 2002)

MONTAGE SOFTWARE TEST PLAN REVISION HISTORY

Description of Revision	Date
Initial Release – Version 1.0	September 24, 2002

Table of Contents

1. Purpose of this Document

The purpose of this document is to describe a test plan for Montage, an astronomical image mosaic service for the NVO. This plan identifies the test platforms and test processes that are needed, describes the test processes that will be performed, the scope and schedule for executing the test plans, describes the scope, approach & methodology, resources & responsibilities and high-level schedule for the testing activities. It will identify the software items and features under test, the test tasks that will be performed and the personnel responsible for each task.

Individual test cases that meet the standards described in this document will be developed and performed for each release according to the schedule given in Table 1. The risks associated with the plan will be tracked in the same way as other project risks, as described in the Montage Software Engineering plan [1].

2. What is Montage?

Montage is an exemplar compute-intensive service for the National Virtual Observatory (NVO). It will deliver on demand *science-grade* astronomical image mosaics that satisfy user-specified parameters of projection, coordinates, size, rotation and spatial sampling. S*cience-grade* in this context means that the impact of the removal of terrestrial and instrumental backgrounds on the fidelity of the data is understood over different spatial scales. The service will deliver mosaics of images released by the 2 Micron All Sky Survey (2MASS), the Digital Palomar Sky Survey (DPOSS) and Sloan Digital Sky Survey (SDSS) projects.

The computing challenge of Montage is to sustain a throughput of 30 square degrees (e.g. thirty 1 degree x 1 degree mosaics, one 5.4 degrees x 5.4 degrees mosaic, etc.) per minute on a 1024 x 400 MHz R12K Processor Origin 3000 *or machine equivalent*. Montage represents an evolution of a baseline engine deployed at JPL, *yourSky*. Incremental deliveries in 2003 and 2004 will progressively improve the science quality, speed and portability of the baseline code. The final deployment will be in January 2005.

Montage will run operationally on the *TeraGrid*, (http://www.npaci.edu/teragrid/). Users will submit requests to Montage through existing astronomical portals, and visualize and interact with the delivered mosaics through the same portals.

A fully documented, portable version of the software will be publicly available for running on local clusters and individual workstations. The compute engine will be quite general and will allow users to build mosaics from their own FITS format data.

3. Test Schedule

Table 1 gives the test schedule for each release of Montage. This schedule includes design and execution of test cases, code updates to correct defects, subsequent regression testing, and final review and sign-off of all test items by the QA officer.

Table 1: Test Schedule for Montage

Milestone & Due Date	Test Case Specification	Test Case Execution	Code Updates	Regression Testing	Sign off by QA
F: 2/28/2003	10/30/2002	1/20/2003	2/1/2003	2/15/2003	2/20/2003
I: 8/15/2003	4/20/2003	6/25/2003	7/25/2003	8/5/2003	8/10/2003
G: 2/28/2004	10/30/2003	1/20/2004	2/1/2004	2/15/2004	2/20/2004
J: 8/15/2004	4/20/2004	6/25/2004	7/25/2004	8/5/2004	8/10/2004
K: 1/10/2005	10/30/2003	12/1/2004	12/15/2004	12/31/2004	1/05/2005

4. Software Test Environment

The Montage project will be responsible for designing and executing test suites that fully exercise the accuracy, stability and performance of Montage. Validation of the code for astrometric and photometric accuracy of the output mosaics will be performed under the guidance of the Customer Review Board. The validators will be scientists who will be given access to the code before public release, in accordance with our policy for customer release (Milestone H).

4.1 Definition of Test Processes

The following three types of test processes will be carried out, defined in Table 2:

- Software testing, including testing of individual components and of the Montage service as a whole
- Installation testing from the ground up; that is, ensuring that dependent third party libraries and Montage itself can all be built and run according to instructions supplied with the code.
- Documentation testing. Montage will be delivered with system and user documentation which must be complete and accurate.

Table 2: Definitions of Test Processes

Test Process	Definition & Scope
Software Testing	
Developer Testing	Testing of individual modules by developers • Test paths through modules • Test interface to module
Functional Testing	Testing of individual modules according to whether they

	satisfy entry and exit criteria
	• Test that operations performed by the modules are done correctly (e.g. output from a coded algorithm is correct)
	• Test interface:
	  o Bad data values (e.g. wrong data type)
	  o Input values out of bounds
	• Test error conditions returned by module
	  o Files cannot be read
	  o Files cannot be created etc…
	
	Functional testing will also be used to develop a series of regression tests that will validate the interface of each module.

Integration testing	Exposes faults between interfaces • Run system as a whole to ensure that interfaces "talk" correctly to each other • correct error conditions are returned when faults between interfaces are found
System testing	Ensures that requirements have been satisfied • All requirements appropriate to that release are met • Traced via Requirements Traceability Matrix
Regression testing	Retest code *on all supported platforms* to ensure that new defects have not been introduced through modifications to other parts of the code. The regression test suite will be built from the results of the developer, functional, system, integration, performance and installation test suites. The regression tests will validate the paths through the code, functionality of the modules and their interfaces, the performance of the system and the fidelity and accuracy of the output mosaics (as specified in the requirements), and the correct installation of the software. Regression testing will be performed via an automated script throughout the project lifetime, following correction for defects and new system builds.
Performance and Stress testing	Ensures that MONTAGE will run at specifications, and to what extent it will perform beyond the limits of its specifications
Beta Testing	Arrange for testing of MONTAGE at a site not involved in development and testing • Performed without a formal test plan; testers use software as they would use it operationally
Acceptance Testing and	Determine the accuracy and fidelity of the image mosaics

Validation	generated by Montage. • Ensure that output mosaics are accurate with respect to astrometry and photometry • Ensure that MONTAGE does not introduce defects into the mosaics • Ensure that Montage correctly handles "bad" or low quality input data Will be largely performed in cooperation with astronomers who are given access to the code before release.
Installation	
Installation testing	Determines that users can download, build and use MONTAGE according to instructions provided with Montage. Includes installation of dependent components.
Documentation	
End User Manual Testing	Document is complete, consistent and error-free. Includes testing of the validation suite that will be delivered with Montage.

4.2 Test Platforms

Successive milestones will emphasize testing on particular platforms, and when Montage is delivered to the community in January 2005, we will have run a complete suite of tests on all the platforms listed in Table 3 below. Testing on IRIX platforms will be performed only as resources permit.

Table 3: Test Platforms and Operating Systems

Machine	OS	Availability
TeraGrid	Red Hat Linux 6.2	Available as TeraGrid Lite. Request for time approved through NVO project
IBM Blue Horizon	AIX 5L	Account at SDSC approved
Linux Cluster	Red Hat Linux 6.2	Account at CACR approved Account at SDSC approved as part of NVO test bed.
Solaris Workstations	Solaris 2.7, 2.8	Available at IPAC
Linux workstations	Red Hat Linux 6.2, 7.x	Available at IPAC, CACR
IPG SGI O2K, O3K	*IRIX 6.5.x*	*Account approved for JPL; used for testing Montage only as project resources permit*

4.3 Test Environments and Test Datasets

Montage will use existing environments for testing. We do not plan to configure or build special test platforms, other than the following standard practices:

- setting up directories on test machines for housing code and dataset;
- installing database client software for loading tables into databases and reading them (vendor supplied or Open Source);
- installing dependent libraries as needed; these are standard astronomy libraries identified in [2];
- Ensuring that the GNU "gcc" compiler is loaded on the test platform

This policy allows to us to test Montage under conditions that closely replicate those in which customers will use it, and eliminates hardware purchase costs.

Montage will use image collections publicly released by 2MASS, DPOSS and SDSS as test data sets. A description of these datasets and their disposition is given in [1]. We also expect that customers will input their private image collections, assumed compliant with the FITS standard. Ensuring that Montage will process these collections is an important part of our test effort, and we will seek the guidance of the review board in identifying potential data sets. This work will be absorbed into our validation plan, and will be formally called out in our test cases as data collections are identified. The remainder of this plan will lay out the test plan for the 2MASS, DPOSS and SDSS data collections.

There are two classes of test environments required to support testing: grid or supercomputer environments, used for operations, and users' local environments. Each environment requires access to processors, data sets, and a database (optional for users performing their custom background rectification; they may input flat file containing the necessary parameters).

Grid and Supercomputing Environments

- **The Teragrid**
    - Hardware: Now under development. SDSC have provided a Linux cluster at SDSC for testing of NVO compute intensive processes, which we will use initially. This cluster ("Teragrid Lite") will be part of the larger Teragrid.
    - Database – SDSC are committed to providing database for this cluster by 12/2002; it will most likely be Sybase. This delivery date will not affect testing Milestone F, as is emphasizes accuracy in the mosaic engine, rather than speed; background rectification can be performed only through reading a flat file, if necessary.
    - Image collections – 2MASS: 4 TB of public data accessible through HPSS, and SDSC now replicating on spinning disk; latter are preferred for testing, but access through HPSS is acceptable (as now done by *yourSky*). DPOSS: available through HPSS at CACR; will be replicated at SDSC for access by

NVO software. SDSS: NVO will negotiate with SDSS to replicate their data at SDSC; in the worst case, we will download public images from the SDSS archive and replicate them manually at SDSC.

Workstation Environments

Montage will support the two most common platforms used in astronomy, Solaris 2.7/2.8 and Linux 6.x and 7.x.

- Solaris
  - Solaris 2.7/2/8 Ultra 10 workstations at IPAC
  - Database: Informix
  - Image Collections: 2MASS quick look (browse) images for 47% of sky archived at IPAC; replicate subsets of 2MASS image collection from SDSC, subset of DPOSS from CACR; SDSS images from project archive
- Linux work stations
  - Linux 6.2, 7.x PC's at CACR
  - Database: MySql
  - Image Collections: replicate subsets of 2MASS Atlas and Quick look images from IPAC and SDSC; full DPOSS collection; SDSS images from project archive.

While the public image data sets from 2MASS, DPOSS and SDSS will be available for test purposes, it is likely that a subset of them will be identified as especially valuable for testing Montage. With the help of the data providers, we will identify images that are particularly "difficult" to mosaic, because for example they contain an unusually high proportion of "bad" or low-quality data. We may modify delivered data to create dedicated "data files from hell" used to test Montage's ability to handle exceptions and out-of-bounds data. A subset of these data will be delivered with the Montage system documentation as a validation test data set. The users' guide will clearly describe whether delivered test data are modifications of publicly released data.

5. Structure of the Test Plan

Montage consists of two engines [2]:

- A background rectification engine, used to generate parameters that will correct individual images for terrestrial and instrumental backgrounds.
- A mosaic engine that performs reprojection, resampling, coordinate transformations, image reprojections, co-additions and constructs the final mosaic.

The components of these engines are shown in Figure 1 and described in Table 4. A key feature of each engine is that *all* its components are stand-alone, and we will therefore develop a test suite for each component *as if it were a software system in its own right*,

with teach suite executed for the platforms listed in Table 3. Following testing of the individual components, we will test the rectification engine and the mosaic engines at the system level by running the processing through the executive modules identified below; these executives are simple front-ends whose sole purpose is to make the calls to processing components, and return messages about the processing to the user. Indeed, the background rectification component can be considered as three sub-components, each run by its own executive, for processing overlaps, differences between pairs of images, and for removing the background itself.

Figure 1: The Design Components of Montage

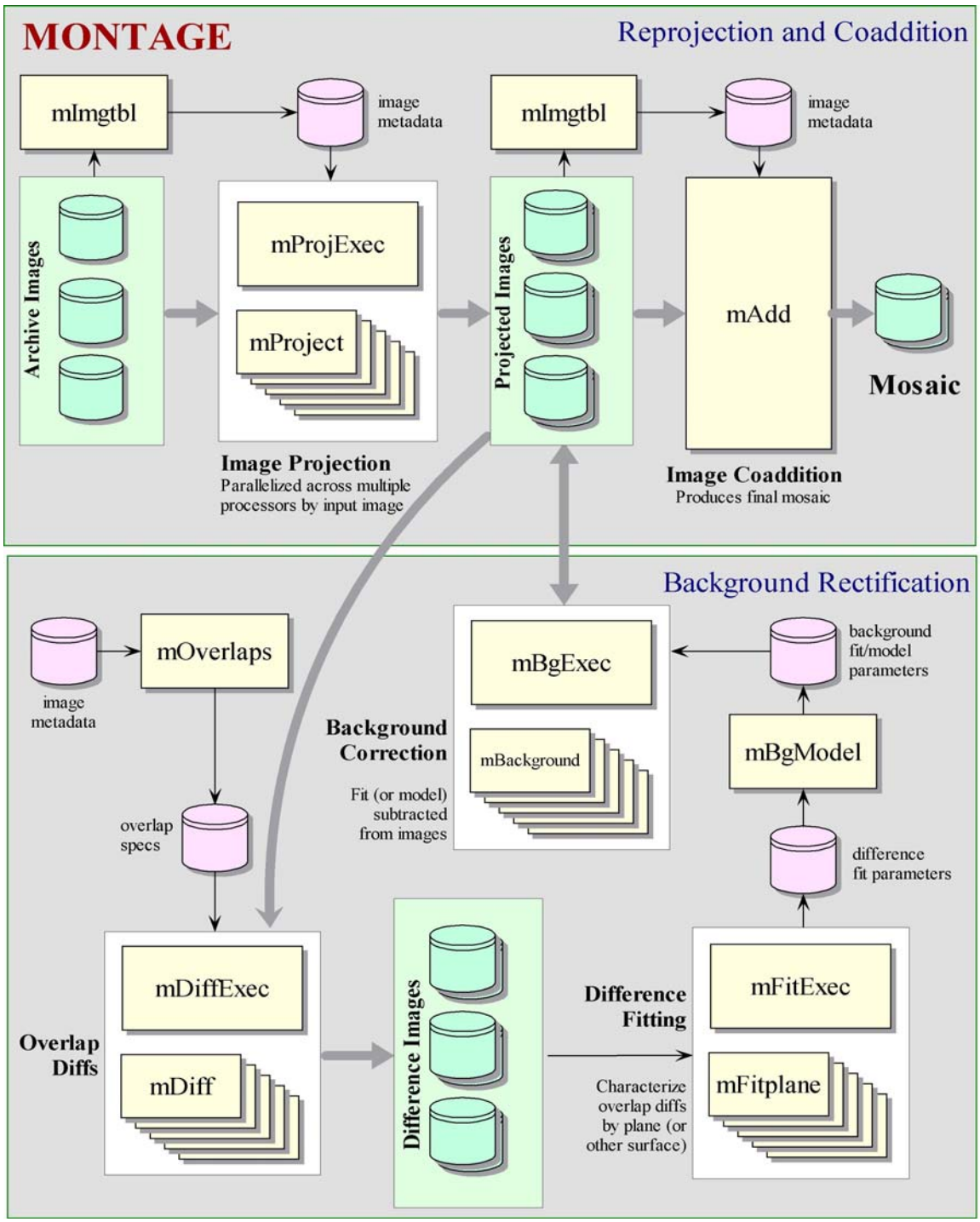

Table 4: Descriptions of the Design Components of Montage

Components run through a common executive are shown with a common background color

Component	Description
Mosaic Engine Components	
`mImgtbl`	Extracts the FITS header geometry information from a set of files and creates an ASCII image metadata table from it used by several of the other programs.
`mProjExec`	**A simple executive which runs `mProject` for each image in an image metadata table.**
`mProject`	Reprojects a single image to the scale defined in a pseudo-FITS header template file (an ASCII file with the output image header lines, but not padded to 80 characters and with new lines at the end of each line). Actually produces a pair of images: the reprojected image and an "area" image consisting of the fraction input pixel sky area that went into each output pixel.
`mAdd`	Coadd the reprojected images using the same FITS header template and working from the same `mImgtbl` list.
Background Rectification Components	
`mOverlaps`	Analyze an image metadata table to determine a list of overlapping images.
`mDiffExec`	**Run `mDiff` on all the pairs identified by `mOverlaps`.**
`mDiff`	Perform a simple image difference between a single pair of overlapping images. This is meant for use on reprojected images where the pixels already line up exactly.
`mFitExec`	**Run `mFitplane` on all the `mOverlaps` pairs. Creates a table of image-to-image difference parameters.**
`mFitplane`	Fit a plane (excluding outlier pixels) to an image. Meant for use on the difference images generated above.
`mBgExec`	**Run `mBackground` on all the images in the metadata table**
`mBgModel`	Modeling/fitting program which uses the image-to-image difference parameter table to interactively determine a set of corrections to apply to each image to achieve a "best" global fit.
`mBackground`	Remove a background from a single image (planar has proven to be adequate for the images we have dealt with).

5.1 Montage Test Suites

We will develop complete test suites for the components identified in Table 4. Broadly speaking, those components that perform the processing will be subject to developer, installation and functional testing, and integration, system regression, performance, beta and validation testing will be performed at the validation level. The Montage user

documentation will describe individual components and the system as a whole, and so documentation testing will be performed for all components.

One exception to the above rule will be made for the reprojection component, mProject. Because it assumes the great bulk of the processing burden, and because it will in many cases be used as a general reprojection tool, we will perform load, beta, and validation testing on it. Table 5 lists the test processes that will make up individual test suites. Each test suite will consist of a sequence of test cases that will fully establish the accuracy, performance and robustness of that component of Montage

The Montage requirements demand that image mosaics can be ordered through web portals and can be visualized through web browsers. Montage will use existing portals and services to satisfy these requirements, and such new services and portals as are developed as part of the NVO project. Testing of data ordering and visualization will be performed as part of system and acceptance testing of the appropriate deliveries.

The first code improvement milestone, F, will emphasize accuracy on Linux 6.x, which will be the OS for the Teragrid and one of the commonest platforms on which Montage will be run. Once the code and algorithms have been validated on Linux, the test results for a given image collection will make up a template that can be used on other platforms.

Table 5: Identification of Test Suites for Montage and Its Components

Components run through a common executive are identified by a common background color

Component	Test Processes†
mImgtbl[1]	S, VN, E, R^2
mProjExec	D, F, I,S, P,B, V, N, E, R^2
mProject	D, F, P, B,V, E, R^2
mAdd	D, F, N, E, R^2
mOverlaps	D, F, S, V, N, E, R^2
mDiffExec	D, F, I,S, P,B, V, N, E, R^2
mDiff	D, F N, E, R^2
mFitExec	D, F, I,S, P,B, V, N, E, R^2
mFitplane	D, F, N, E, R^2
mBgExec	D, F, I,S, P,B, V, N, E, R^2
mBgModel	D, F, N, E, R^2
mBackground	D, F, N, E, R^2

† Key to Test Processes: D= developer; F= functional; I= integration; S= System; P= performance and stress; B=beta; V=acceptance and validation; N= installation; E= End User Documentation; R=regression

[1] mImgtbl has been in operational use at the Infrared Science Archive since 2000. Because it is robust and well-tested, we will not perform developer and functional testing here, but we will test it as part of integration, system and acceptance testing. Such updates to mImgtbl as are incorporated into IRSA will be migrated to Montage: we plan to have only one operational version.

[2] When needed after defects have been corrected or after a new system build.

5.2 Entrance and Exit Criteria for Test Suites

For each suite:

Entrance criteria: Collection of documented test cases approved the PI and the QA officer, committed to CVS. Each test case will include a detailed description of the expected results, such as technical specifications of the image mosaics output from Montage, and descriptions of the error messages expected to be generated when testing failure modes. Test cases will follow the structure given in Section 5.

Exit criteria: A successful test will replicate the expected outcome. Where this criterion is not met, the Montage team will decide on the response to each defect. Following correction of defects, appropriate retesting and regression testing will be performed. Defects not corrected for a particular release will be fully documented in the User Documentation.

Regression Tests: Regression testing must be performed throughout the project, following correction of defects and system builds, including builds that will support incremental releases outside the major milestones. The regression test suite will be built from the results of the functional, system, integration, performance and installation test suites. The results will validate the functionality of the modules and their interfaces, the performance of the system and the fidelity and accuracy of the output mosaics (as specified in the requirements), and the correct installation of the software.

The regression test suite will consist of test cases successfully completed from functional, integration, system and performance tests. These tests must fully exercise Montage's functionality, robustness, interfaces and performance. The exit criterion for regression testing is that the output of each test case must be identical in successive test runs, except in those cases affected by a corrected defect: in these cases, the affects of the bug on the output must match predictions.

For an incremental release, a complete set of regression tests that thoroughly exercise Montage must be performed. For corrections to defects during a development cycle, a partial set of tests may be run. While this will be assessed on a case-by-case basis, the exit criterion will remain as above.

Documentation Tests: Some special remarks are applicable here. The documentation will include items such as algorithm descriptions, where a quantifiable test outcome cannot be specified. The Customer Review Board will review such documentation, and their explicit approval of the quality of the documentation will be considered as the exit criterion. Otherwise, documentation testing will be performed as per other tests. For example, build instructions will be accompanied by a description of the characteristics of the expected executable images (number, size …). Testing will involve building Montage

according to the instructions set forth in the documentation, and ensuring that the executable images must have the characteristics described.

5.3 Test Plan Scope and Schedule

Montage will be delivered incrementally between February 2003 and January 2005. Successive deliveries will alternately emphasize accuracy and improvements in performance on the one hand, and improvements in portability and the number of image data sets supported on the other. The matrix in Table 6 summarizes those test suites that will be performed on each delivery.

Table 6: Scope and Schedule for Montage Testing

Milestone, Due Date & Version	Technical Summary of Milestone	Platforms Supported	Data Collections and Scope of Testing†
F: First Code Improvement; 2/28/2003; 1.0	Accuracy and robustness over performance and interoperability.	Linux 6.x on Teragrid, Linux 7.x workstations	2MASS: All test suites (except P and portals and visualization in S, V)
I: Interoperability Prototype; 8/15/2003; 1.x	Use code in F): Performance comparison Order mosaics through extant clients	Linux 6.x on Teragrid. Linux 7.x	DPOSS: All test suites 2MASS: R, P, portals & visualization in S, V.
G: Second Code Improvement; 2/28/2004; 2.0	Performance improvement on Teragrid. Deployment of cache.	Linux 6.x, Teragrid. Linux 7.x Solaris 2.7/2.8	2MASS, DPOSS: R,and P on Linux, full test suites on Solaris SDSS: Complete test suites on Linux 6.x (Teragrid), Linux 7.x, Solaris
J: Full interoperability; 8/15/2004; 2.x	Full interoperability with 2MASS, DPOSS, SDSS data sets. Full toolkit for image manipulation. .	Linux 6.x, Teragrid. Linux 7.x Solaris 2.7/2.8 AIX	2MASS, DPOSS, SDSS: R and visualization in S,V for Linux, Solaris. Full test series on AIX

K: Customer Delivery; 1/10/2005; 3.0	Final performance milestone.	Linux 6.x, Teragrid. Linux 7.x Solaris 2.7, 2.8 AIX *IRIX 6.5*[1]	2MASS, DPOSS, SDSS: R, P all platforms except IRIX 6.5 *Full test series on IRIX 6.5*[1]

† Key to Test Processes: S= System; P= performance and stress; V=acceptance and validation; N= installation; R=regression

[1] Only as resources permit

6 Test Approach and Methodology

6.1 Drivers and Test Tools

Generally, we will execute test cases by running Montage from the command line. We anticipate that some parts of the testing will benefit from automation, but we will restrict drivers to simple Unix scripts.

6.2 Specifications of Test Cases

For each test suite, we will define a series of test cases that will be performed for all supported platforms. The definitions of the test suites and the test reports will be archived under CVS, following approval by the PI and the QA officer. All test cases/reports will have the structure described in Table 7, and will be documented on the Montage project internal web page. Through the life of the project, the test cases will be run on all platforms supported by Montage, and results for each platform will be documented as links from the web page for that test case.

Table 7: Structure of Test Cases

Test Case Item	Description
Identifier	Unique identifier to reference the item under test, the platform, the test process, and the test number *e.g. mProjec_F_005 denotes the 5th test case of Functional testing of mProject.*
Purpose	The aim of the test, including the name of the module and/or the feature in the module under test. Will include references to product specs or design docs as necessary.
Montage Version; Module Version	Version of Montage system; version of

	module under test (from CVS)
Test case dependencies	Identification of other test cases that may affect the result of this case & why
Tester	Name of tester
Date	Date(s) when tests performed
Input Specification	All inputs and conditions that are needed to run the test
Output Specifications	Expected Results
Procedure	Procedure to follow
Test Conditions	Machine, OS etc on which test was run
Pass/Fail criteria	Precise description of pass/fail criteria.
Test Results	Describe test results vs. what was expected
Pass/Fail	Pick one; Bug ID if Fail
Defect Severity	Fatal (correct immediately); Serious (must correct for release); Cosmetic or
Retest Results	Results of retest if F; state new module numbers (if updated).

6.3 Test Case Tracking

We will use a simple web form that will be available on the project internal web site for tracking the status of each test case. The QA officer will be responsible for determining the status of the test cases and updating the test case status page.

The page will be organized according to the test processes that must be performed for each milestone, and will have the structure shown in Table 8.

Table 8: Sample Test Case Tracking

Milestone F: First Code Improvement; Platform: RedHat Linux 7.3			
Test Suite/Cases: tImgTbl_F	Test Date; P/F	Bug ID	Retest Date P/F
001	10/15/2002; P		
002	10/15/2002; F	25, 26	11/30/2002; P
. . .			

6.4 Quality Assurance Tools

The QA tools described in this subsection are requisite to the test plan and were described in [1]. They are repeated here for completeness.

6.4.1 Requirements Traceability Matrix

This matrix will be posted on the project internal web site and will be used to ensure full requirements coverage. The matrix will be an Excel spreadsheet table with column entries as in Table 9, with one entry for each requirement with a (fictitious) sample entry

Table 9: Requirements Traceability Matrix

Req. Spec. No.	Req. Statement	S/W module	Test Spec.	Test Case #	Verification	Mod. Field
10	Support coadditions through simple averaging and medians	Do_co_adds.c	Developer,functional, integration, acceptance, beta	16,19-22, 34,66-69	Fully verified	Version 2.0 requirements review – added median co-adds
etc						
etc						

The interpretation of the columns is as follows:

Column	Description	Timeline
Requirements Specifications Number	The requirement paragraph number as listed in the Requirements Specification document	initial requirements analysis
Requirement Statement	Paraphrase of the actual requirement as it appears in the Requirements Specification document	initial requirements analysis
Software Module	The module/subroutine that addresses the requirement	Detailed design phase
Test Spec.	The Test plan that contains the test case/procedure that validates the requirement. e.g., unit test plan, Integration test plan, Acceptance test plan	Test Plan development
Test Case #	The test cases that will be run to verify the requirement	Test Plan development
Verification	How well the requirement was verified: not verified; partially verified; fully verified	after executing the test procedure
Modification Field	Used in case requirement has been modified in any way throughout the life of the project. Indicate disposition (changed / eliminated / replaced), and authority for modification, e.g., eliminated – Requirements review, 10/17/01	Throughout the project as requirements are modified

6.4.2 Defect Tracking

The project will deploy a defect tracking system to records and track defects reported in reviews, testing and reported by defects. Defect tracking will take place throughout the project, beginning with initial design review. The defect tracker will report the following information:

Identifier
Description
Priority
Assignee
Montage version number
Development Phase (requirements, design, …, user reported)
Status (Either open or closed)
Reporter
Platform (OS, CPU, etc.)
Submission Date
Close-Out Date
Resolution

Staff Hours to Resolve Defect

We are currently evaluating a commercial product, TestTrack, offered by Seapine Software, Inc. We expect to make a decision on a defect tracker by the end of September 2002.

6.4.3 Software testing metrics

The following metrics are applicable to testing, and have been adapted from those described in [1]. The QA officer will be responsible for generating and publishing the metrics. As testing proceeds, he or she will generate and update these metrics and publish them on the internal web page. Following release, the metrics for that version will be published on the public web page. They will be generated by the QA officer, and published on the Montage web page for each release.

Software Complexity
- McCabe metric or equivalent for each delivered module
- Changes in this metric for successive releases (except for first code improvement)

Schedule
- Number of FTE equivalents performed to complete the test plan for each milestone (actual vs. planned)

Software Quality
- Number of defects reported per release in:
  - Reviews
  - Testing
  - Operations (reported by users)
  - Level of effort to correct each defect
  - Length of time the defect was "open"

- Number of defects, ordered by priority, open as function of time.

7 Resources and Responsibilities.

The development of Software test cases and its execution will be led by the Montage Project Manager and the JPL Line Manager, and will be assisted by the QA officer. None of these persons will participate in code development. They will be supported by users who will largely help with the validation of Montage.

Table 10 summarizes the roles and responsibilities of the Montage staff in the designing and executing the test plan.

Table 10: Test Responsibilities on Montage

Task	Project management	Developers	Quality Assurance	Customers	Tech Writer	Users
Develop Test Cases	X					
Matrix Test Suites and Test Cases to Deliveries	X					
Review Test Matrix			X			
Maintain Requirements Traceability Matrix			X			
Examine and approve test reports; commit to CVS			X			
Maintain test tracking matrix			X			
Evaluate Defect tracker	X	X	X			
Organize user validation effort	X					
S/W metrics maintenance			X			
Get user buy-in to validation plan	X			X		
Prepare installation and build guides, users guide, document test validation suite.					X	
Developer testing		X				
User testing - validation						X
End user documentation						X
All other testing	X					
Develop test data sets	X	X				X
Develop validation suite for delivery with code	X	X				
Decisions of fixing reported defects (bug committee)	X	X				
Documentation of open bugs for each milestone in user guide			X			
Organize beta testing	X			X		
Ensure test machines are available	X	X				

References

[1] "Software Engineering Plan for Montage". Version 1.0 (May 31, 2002);
http://montage.ipac.caltech.edu/Documentation/MONTAGE_SEP.doc

[2] "Software Design Specification for Montage". Version 1.0 (June 30, 2002)
http://montage.ipac.caltech.edu/Documentation/Montage_Design.doc

Acronyms

2MASS	Two Micron All Sky Survey
AIX	Advanced Interaction eXecutive
ANSI	American National Standards Institute
ASCII	American Standard Code for Information Interchange
CACR	Center for Advanced Computing Research
DBMS	DataBase Management System
DPOSS	Digital Palomar Observatory Sky Survey
FITS	Flexible Image Transport System
FTE	Full Time Equivalent
GNU	Gnu's Not Unix
HPSS	High Performance Storage Server
IPAC	Infrared Processing and Analysis Center
IPG	Information Power Grid
JPL	Jet Propulsion Laboratory
NVO	National Virtual Observatory
PC	Personal Computer
PI	Principal Investigator
SDSC	San Diego Supercomputer Center
SDSS	Sloan Digital Sky Survey
SGI	Silicon Graphics Inc
SRB	Storage Resource Broker
TB	TeraByte