Bioconductor / caBIG Task 2.6.1

Test Plan

Version 1.2

[*Insert approval date of document*]

Document Change Record

Version Number	Date	Description
1.0	24 August 2006	Draft
1.1	20 September 2006	Final
1.2	29 September 2006	Revised final; Architecture comments

TABLE OF CONTENTS

1. INTRODUCTION

This Test Plan prescribes the scope, approach, resources, and schedule of the testing activities. It identifies the items being tested, features to be tested, testing tasks to be performed, personnel responsible for each task, and risks associated with this plan.

1.1 SCOPE

This document provides instruction and strategy for incorporating Software Testing practices and procedures into the Bioconductor / caBIG project. This document demonstrates the application of testing on this project and provides guidelines for implementing procedures supporting this function.

Bioconductor is a collection of open-source software components based on the R programming language. Bioconductor is used for gene expression and other high-throughput analysis in molecular biology. R packages are collections of algorithms grouped to facilitate particular analyses. The software within the scope of this document allows R package developers to expose the functionality of their package as analytic services on caGRID. Our *primary concern* is the development of tools for converting existing Bioconductor packages to caGrid analytic services. Our *secondary concern* is to expose exemplar functionality as caGrid analytic services.

1.1.1 Identification

This document refers to Bioconductor / caBIG version 1.0.

1.1.2 Document Overview

This Test Plan defines the strategies necessary to accomplish the testing activities associated with Bioconductor / caBIG. Testing procedural instructions are included in the Standard Operating Procedures (SOPs). The remaining Test Plan sections are organized as follows:

- **Section 2: Strategy**: Describes the overall approach and techniques to be used during testing.
- **Section 3. Software Test Environment**: Describes the intended testing environment.
- **Section 4. Test Identification**: Identifies and describes each of the tests.
- **Section 5. Test Schedules**: Contains or references the schedules for conducting testing.
- **Section 6. Requirements Traceability:** Identifies traceability from this Test Plan to the Requirements.
- **Section 7. Risks**: Identifies the risks associated with the Test Plan.
- **Section 8. Notes:** Contains general information that aids in the understanding of this document.

- **Appendix A. Acronym List:** Defines the acronyms used on the project.
- ***[Appendix B. Forms and Templates:*** *Contains the forms and templates that will be used on the project.]*
- *[**Other appendices**: Provides any additional information published separately for convenience in document maintenance.]*

1.2 RESOURCES

Participants involved in creating the Project Test Plan include the Bioconductor / caBIG developers at the Fred Hutchinson Cancer Research Center, and the Adopter group located at Northwestern University. Responsibilities of these groups are outlined below.

1.3 REFERENCED DOCUMENTS

For additional project specific information, refer to the following documents:

- Bioconductor / caBIG Software Requirements and Specifications[1]
- Bioconductor / caBIG Use Case documents[2]
- Bioconductor / caBIG UML Models[3]

[1] http://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/bioconductor/Developer_FHCC/Task%202.4.2_Final%20Req%20and%20Spec%20Document/?cvsroot=bioconductor

[2] http://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/bioconductor/Developer_FHCC/Task%202.2.2_Final%20Use%20Case%20Document/?cvsroot=bioconductor

[3] http://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/bioconductor/Developer_FHCC/Task%202.7.2_Final%20UML%20Models%20for%20Selected%20Bioconductor%20Packages/?cvsroot=bioconductor

2. SOFTWARE TEST STRATEGY

The overall software test strategy is to employ ongoing regression and integration testing during initial phases of development. As software becomes stable and comprehensive, additional system, and performance tests are performed. User acceptance testing is the responsibility of the project adopter.

2.1 OBJECTIVES

The objective of the test plan is to identify integration, system, and unit testing approaches to ensure deployment of a fully functional Bioconductor / caBIG v 1.0. The test plan outlines minimal performance expectations. User acceptance testing is conducted by the adopter, in consultation with the developer. Specific objectives include:

- Assure that Bioconductor / caBIG meets all approved requirements and specifications.
- Identify and record for future action all risks and issues arising during testing.
- Identify product development and user acceptance enhancements for incorporation in the initial or future software releases.

2.2 APPROACH

The following table summarizes key parts of the test strategies; textual description follows.

	Type of test		
Requirement	Unit / functional	Integration	System
caBIG			Conformance to caDSR, EVS, caCORE API to caGrid analytic service requirements
RWebServices	White-box Java class and package generation.	R / Java interface. Web and analytic service generation.	
Exemplars	R-level branch coverage and white-box evaluation.	Exception handling. Web service function, local client, performance assessment	Black-box service evaluation

2.2.1 caBIG Requirements

caBIG requirements of Bioconductor / caBIG include vocabulary and data element conformance to caDSR and EVS, caCORE-like API access, and operation as caGrid analytic service. Testing these requirements primarily involve system and integration tests. Additional concerns about large-data throughput encourage performance testing.

Availability of exemplar functionality as published caGrid services requires conformance to caDSR and EVS standards, the ability to provide caCORE-like API access, and web service interfaces to be a fully compliant caGrid analytic service. Hence successful deployment and invocation of exemplar services on caGrid must include system and integration tests that these requirements have been met.

Adequate branch coverage of these requirements involves testing for correct return values, and the generation of exceptions in response to internal error conditions during service evaluation. This coverage will be obtained by white-box testing using input data known to generate exceptional conditions.

Performance will be evaluated using white-box methods that identify ‘typical’ sized data objects, and measure throughput and resource use during service invocation. Performance tests will specifically exclude requirements associated with receiving, decoding, and encoding the SOAP-based service request, focusing exclusively on stages when RWebServices and the functionality identified for this project handle, process, or exchange data. Tests will be graded as ‘passing’ performance testing when the function takes no more than twice the equivalent execution time and no more than twice the system resources (physical and virtual memory, CPU time) as that function run exclusively in R, independent of time required to serialize and deserialize SOAP messages and to instantiate the function..

2.2.2 RWebServices Requirements

RWebServices requirements involve production of Java class representations of R data and methods, and method dispatch and receipt of service requests, including error recovery.

Evaluating RWebServices’ ability to generate analytic service interfaces involves functional and structural test approaches. A primary test type will be functional tests, comparing RWebServices generated service interfaces with hand-curated output. Conformance test cases will be designed to evaluate multiple branches of RWebServices, including correct generation corresponding to services of individual and multiple packages, generation of Java data structures corresponding to the full range of ‘native’ R data types, and generation of Java data structures corresponding to complex S4 objects relevant to the domain models of the exemplar packages.

Functional and structural methods using integration, system, and regression tests will be performed on the R / Java interface. Test suites will ensure correct round-trip data transfer between R and Java. Tests will involve R native, Java native, and S4 objects. Test data objects will include supported IEEE numerical concepts such as NaN, and Inf, the R concept of NA (signaling an error when present in return values), and invalid object transfer. The diversity of object types provides extensive path coverage. Branch coverage is assured by the diversity of object types, including invalid object transfer.

Automated unit, functional, structural, integration and system testing will invoke RWebServices operating on a scripted server using JUnit tests. These tests will include requests that emulate client-side requests for web analytic services. Components involved in these tests include RWebServices and the exemplars, in addition to the SJava package and R program. White-box methods will be used to evaluate exception and error handling.

2.2.3 Exemplar Requirements

Exemplar requirements are that the data and methods of the exemplar be exposed as web services, and that service invocation with domain- or method-specific objects and tuning parameters return appropriately structured return objects. The exemplars are also required to signal error or exception conditions.

Testing of exemplar functionality at the R level will include functional testing to test returned values with known white box data sets and structural testing approaches designed to provide extensive path and branch coverage. Regression tests will be written as bugs are identified to prevent re-injection of the bugs into the code base. Path and branch coverage, including handling exceptions, will be evaluated using data instances designed for these purposes.

System and integration tests of exemplar functionality will include the same test suites as the unit tests described above, but will be run though RWebServices.

Performance tests of exemplar functionality will assess ability of functions invoked through RWebServices to fulfill service requirements in no less than 2x the time required for servicing comparable requests at the native R level, and using no more than 2x the resources required for native requests.

2.3 DESCRIPTION OF FUNCTIONALITY

Functionality being evaluated is as specified in the Software Requirements and Specifications document[4].

2.4 SPECIFIC EXCLUSIONS

The Bioconductor / caBIG software relies on the correct prior operation of additional software components, and the testing procedures here assume the correct installation and functioning of these existing components. Specific components are as follows. The grid services infrastructure, including Apache, Tomcat, and the Globus toolkit, the Java programming language, the R

4 http://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/bioconductor/Developer_FHCC/Task%202.4.2_Final%20Req%20and%20Spec%20Document/?cvsroot=bioconductor

software for statistical analysis[5], and underlying R / Bioconductor software packages[6] required for service fulfillment, and the identification or remediation any errors/bugs/inconsistencies in these components are outside the scope of this project.

The Bioconductor / caBIG software relies on specific functionality of the additional components outlined above. Key additional component functionality lying outside the scope of the testing procedures outlined here include the ability to formulate, send, and receive appropriately structured requests for analytic services, the ability of the R environment to access packages required for a specific exemplar, and to obtain sufficient system resources and permissions for secure evaluation.

2.5 DEPENDENCIES & ASSUMPTIONS

The dependencies and assumptions include those outlined in the previous section (viz., ability to provide analytic services and to perform calculation in the R software environment).

Testing the portion of Bioconductor / caBIG that involves exposing exemplar services[7] assumes familiarity with the caGrid environment, including detailed understanding of caGrid architecture and installation procedures. Testing procedures required for exposing services in addition to the exemplars[8] provided by Bioconductor / caBIG requires detailed understanding of caBIG procedures for semantic annotation, in addition to detailed understanding of underlying service functionality.

2.6 GENERAL CRITERIA FOR SUCCESS

General criteria for success are as follows:

- All tests complete
- Error conditions and other exception handling tests will be considered to pass if the test results in the proper exception/error signal
- Minor defects may be present that limit data types or other aspects of service invocation.
- Minor defects in exemplar functionality resulting in incorrect calculations will be tolerated if identified as exceptions.

---

[5] http://www.r-project.org/

[6] http://www.bioconductor.org/

[7] Use case 3.3, http://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/bioconductor/Developer_FHCC/Task%202.2.2_Final%20Use%20Case%20Document/?cvsroot=bioconductor

[8] Use cases 3.1 and 3.2, http://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/bioconductor/Developer_FHCC/Task%202.2.2_Final%20Use%20Case%20Document/?cvsroot=bioconductor

- Substantial components of code will be covered, especially in assessing exemplar functionality and RWebServices performance during service invocation.

2.6.1 Readiness Criteria

Work will be ready to be tested following completion of the test plan, development of test data set and test scripts, and realization of the test environments. Unit tests will be employed throughout out product development. Readiness of software components for system and integration tests requires functional (i.e., tested) RWebServices and exemplar components.

2.6.2 Pass/Fail Criteria

Pass / fail criteria will involve test execution in conjunction with assessment of test outcomes as consistent with requirements and specifications.

2.6.3 Completion Criteria

The criteria for completion of the testing procedures is that the system produces the output specified by the test within expected performance requirements. Testing is considered completed when:

- The assigned test scripts have been executed.
- Defects and discrepancies are documented, resolved, verified, or designated as future changes.

2.6.4 Acceptance Criteria

Criteria for acceptance of completion of tests will involve successful test completion coupled with available test logs and documentation of test performance.

3. SOFTWARE TEST ENVIRONMENT

This section describes the software test environment at each intended test site.

3.1 DEVELOPER TEST ENVIRONMENT: FRED HUTCHINSON CANCER RESEARCH CENTER

The developer test environment is a stable area for independent integration, system, and regression testing for use by the developer testing team. The area is populated by objects as they are completed by developers. The environment ensures that objects are tested with supporting software components consistent with requirements and specifications of Bioconductor / caBIG. The test environment is repopulated with revised products of Bioconductor / caBIG as appropriate.

3.1.1 Software Items

Base software items for the test environment include system software (SuSE 9.2), R 2.4.0, Bioconductor base packages 1.9, Java 1.5.0, SJava 0.64, gcc 4.0.2 and related libraries, Ant 1.6, Globus 4.0.2.

Software items for test performance include R CMD facilities in R, RUnit 0.4.12, and JUnit 4.0 and other tools employed in an *ad hoc* basis.

Bug tracking will use the gForge bug tracker.

3.1.2 Hardware and Firmware Items

Testing platforms include 32- and 64-bit i386 and x86_64 hardware.

3.1.3 Other Materials

Other major components include sample data sets coupled with test instructions.

3.1.4 Participating Organizations

Regression, integration, and systems tests run at this site are performed by project developers.

3.2 ADOPTER TEST SITE

The Adopter test site provides a near-production environment for the client acceptance testing. Bioconductor / caBIG software components are released by the Developer and managed by the Adopter. The Adopter is responsible for software, hardware, and other aspects of adopter test site configuration. The adopter test site also tries to use the current stable release of all software suites that are publicly available.

3.2.1 Software Items

Base software items for the test environment include system software (RHEL4-based caGRID 1.0), R 2.4.0, Bioconductor base packages 1.9, Java 1.5.0, SJava 0.69, gcc 4.1.0 and related libraries, Ant 1.6, Globus 4.0.3.

Software items for test performance include R CMD facilities in R, RUnit 0.4.12, and JUnit 4.1 and other tools employed in an *ad hoc* basis.

Bug tracking will use the gForge bug tracker.

3.2.2 Hardware and Firmware Items

Testing platforms include 32-bit i386 and 64-bit x86_64 hardware.

3.2.3 Other Materials

Test data and scripts will be provided by the adopter. Other major components include sample data sets coupled with test instructions.

3.2.4 Participating Organizations

Regression, integration, and systems tests run at this site are performed by adopters.

4. TEST SCHEDULES

Phase 1: Integration Testing
Duration: September 2006 – March 2007

Phase 2: System Testing
Duration: November 2006 – March 2007

Phase 3: User Acceptance Testing
Duration: Determined by adopter

4.1 TIME FRAMES FOR TESTING

The Test Manager will coordinate with the Project Manager and add the planned testing activities to the master project schedule.

5. RISKS

Risks related to testing are identified in the project's Risk Matrix[9].

[9] http://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/bioconductor/Developer_FHCC/Task%202.1_Risk%20Management%20Matrix/?cvsroot=bioconductor

APPENDIX A – ACRONYM LIST

Acronym	Description
CCR	Change control request
HSTOMP	Health Services Team Organizational Management Portal
MS	Microsoft
PM	Project Manager
RM	Requirements Manager
RTM	Requirements Traceability Matrix
SCM	Software Configuration Management
SDLC	Software Development Lifecycle
SE	Software Engineering
SEPG	Software Engineering Process Group
SM	Software Manager
SOP	Standard Operating Procedure
SPI	Software Process Improvement
SQA	Software Quality Assurance
SW	Software
TM	Test Manager
VM	Version Manager

APPENDIX B – TEST REPORT (TEMPLATE)

Test Overview

[Provide a brief overview of the tested software and summarize the testing effort noting the release version, number of test phases, number of rounds of tests executed within each test phase and the overall duration of the entire testing effort. For example, this report documents the results of the xx.xx release testing effort which consisted of xx CCRs. Testing was divided into 3 phases: Phase 1: Integration Testing, Phase 2: System Testing consisting of 2 rounds and Phase 3: User Acceptance Testing (UAT) consisting of 2 rounds. The entire duration of all combined testing phases was approximately 5 months.]

[Note the duration of each test phase by round below]

Phase 1: Integration Testing
Duration: September 2006 – March 2007

Phase 2: System Testing
Duration: November 2006 – March 2007

Phase 3: User Acceptance Testing
Duration: November 2006 – March 2007

Defect Summary

Phase 1: Integration Testing

		Opened CCRs During Phase	Resolved CCRs	Unresolved CCRs
Priority	X	0	0	0
	X	0	0	0
	X	0	0	0
	X	0	0	0
	X	0	0	0
Total		0	0	0

Phase 2: System Testing

		Opened CCRs During Phase	Resolved CCRs	Unresolved CCRs
Priorit y	X	0	0	0
	X	0	0	0
	X	0	0	0

		Opened CCRs During Phase	Resolved CCRs	Unresolved CCRs
	X	0	0	0
	x	0	0	0
Total		0	0	0

Phase 3: User Acceptance Testing

		Opened CCRs During Phase	Resolved CCRs	Unresolved CCRs
Priority	X	0	0	0
	X	0	0	0
	X	0	0	0
	X	0	0	0
	x	0	0	0
Total		0	0	0

Test Script Summary

[Note the name of the test script executed, execution date, actual test script run time, pass / fail status for each testing phase in the tables below Additional tables / information can be added for each round of testing. This information can be compiled and generated from testing related software if implemented on your project.]

Phase 1: Integration Testing

Test Script Name	Date Executed	Act Run Time (minutes)	P/F	Comments

Phase 2: System Testing

Test Script Name	Date Executed	Act Run Time (minutes)	P/F	Comments

Test Script Name	Date Executed	Act Run Time (minutes)	P/F	Comments

Phase 3: User Acceptance Testing

Test Script Name	Date Executed	Act Run Time (minutes)	P/F	Comments

Conclusion

[Provide a brief summary of any deviations or problems that may have caused, or contributed to, a deviation from the test approach described in the test plan. Summarize successful practices and issues in the table below. Note possible future mitigating actions for each issue.]

Successful Practices:	
•	
•	
Areas for improvement:	**Possible mitigation action:**
•	•

APPENDIX C – TEST IDENTIFICATION

This section identifies and describes each test to which this Test Plan applies.

PLANNED TESTS

This section describes the total scope of the planned testing.

Item(s) to be Tested

CCR ID	Test Script	Dependency
Insert the identifier of the CCR, or unit(s) being tested	The name of the test script to be executed.	Any technical dependencies related to the specific test script.