Integrating Bioconductor and R into caBIG™

Bioconductor / caBIG™

Software Requirements and Specification Document

Version 1.0

[Insert approval date of document]

Document Change Record

Version Number	Date	Description
1.0	*May 2006*	Draft
1.1	*June 2006*	VCDE comments
1.2	*July 2006*	Architecture, adopter comments
1.3	*August 2006*	Workspace comments

TABLE OF CONTENTS

1. INTRODUCTION

This SRSD captures the complete software requirements for the *Bioconductor / caBIG*™ 1.0 release.

The purpose of this module is to provide tools to expose existing Bioconductor 'packages' as caGrid services.

1.1 SCOPE

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.

This module allows R package developers to expose the functionality of their package as analytic services on caGrid 1.0. Operationally this will be performed at two different, but related, levels. One level is to expose Bioconductor packages on caGrid. A second level is to expose particular pipelines as 'exemplars'. Use cases outlining these goals are available[1].

The *primary concern* of this project is to develop tools for converting existing Bioconductor packages to caGrid analytic services. This is in contrast to other modules, which may focus on providing specific functionality as analytic services. This portion of the project involves development of software components to wrap functionality written in the R programming language as Java components, and to expose these Java components as analytic web services.

The *secondary concern* of this project is to expose particular Bioconductor pipelines as 'exemplars'. Exemplars are chosen from three Bioconductor packages. Candidate packages include *affy* (used for microarray analysis) *PROcess* (SELDI-TOF proteomics data) and *DNAcopy* (array CGH data).

1.1.1 Identification

Bioconductor / caBIG™ 1.0.

1.1.2 System Overview

The purpose of the system and software is to provide tools that facilitate developing existing Bioconductor packages as web services, exposure of web-services enabled packages as caGrid services, and discovery and invocation of web-services enabled packages as caGrid analytic services.

The tools in *Bioconductor / caBIG*™ 1.0 represent new software development, sponsored by the National Cancer Institute through the cancer Biomedical Informatics Grid (caBIG™). End users include caBIG™ participants, as well as members of the public. Module development is

[1] http://cabigcvs.nci.nih.gov/viewcvs/viewcvs.cgi/bioconductor/AdministrativeDeliverables/Use_Case.pdf

conducted at the Fred Hutchinson Cancer Research Center in Seattle, WA, under the overall direction of Dr. Robert Gentleman.

Software developed in this project will be installed and used at our caBIG™ adopter sites (Northwestern University) and will be available for installation and use at other caGrid locations.

1.1.3 Document Overview

This SRSD captures the complete software requirements for *Bioconductor / caBIG*™ 1.0 as part of the *Integrating Bioconductor and R into caBIG*™ project.

There are no security or privacy considerations directly associated with use of this software. However, software underlying the project cannot be considered secure; security must be provided by other caGrid software layers.

Software developed specifically for this project is licensed under caBIG™ licensing terms. The underlying R software is open source and licensed under the GPL version 2. Additional license restrictions apply to individual Bioconductor packages.

The remaining SRSD sections are organized as follows:

- **Section 2. Project Description**: Describes the general factors that affect *Bioconductor / caBIG*™ 1.0
- **Section 3. Requirements**: Describes all the software requirements to a level of detail sufficient to enable designers to design a system to satisfy those requirements, and testers to test that the system satisfies those requirements.
- **Appendix A. Acronym List:** Defines the abbreviations used on the project.

1.2 REFERENCE DOCUMENTS

For additional project and requirement specific information, please consult the administrative deliverables and documentation available at:

- http://gforge.nci.nih.gov/plugins/scmcvs/cvsweb.php/bioconductor/Developer_FHCC/?cvsroot=bioconductor

2. PROJECT DESCRIPTION

2.1 PROJECT PERSPECTIVE

The main software component to be produced by this project is the RWebServices package. In broad terms, the software is a collaboration between three R packages (TypeInfo, RWebServices [to be developed here], and SJava) to allow R functions to be wrapped as Java classes. Script and other caCORE and caGrid methodologies (incorporating existing technologies, augmented with documentation and development here) transform the Java classes to caGrid-enabled components. Additional facilities provided by the RWebServices package facilitate invocation of Bioconductor functions as caGrid analytic services.

The delivery platform being targeted is a current Linux operating system capable of caGrid service; this does not preclude deployment on Windows operating systems. Specific constraints are listed below.

2.2 PROJECT FUNCTIONS

TypeInfo (a previously developed product) is an R package to provide strong typing and introspection on function argument and return types. TypeInfo is used to annotate functions in existing Bioconductor packages, and is a necessary first step in making the functions available as web services.

RWebServices is to be developed here as an R package (with some C code) that produces Java beans encapsulating the function and data type signatures of TypeInfo-annotated functions. The package uses type specification and R introspection facilities to determine the underlying data structure of R objects. The data structure is then translated into a corresponding Java class representation.

RWebServices also facilitates analytic service invocation. It does so by including in automatically generated Java classes the methods required for invoking an R evaluator and calling appropriate R functions from Java, and by providing native translation services between Java objects and their R representation.

SJava is an existing technology consisting of R, Java, and C code. It allows R functions to be called from Java, and vice versa. SJava recognizes Java class objects, and translates data represented in Java classes to their corresponding R representation. Data translation occurs at the C level. The Java classes produced by RWebServices are fully interoperable with SJava.

To be available as caGrid analytic services, standard web services require semantic annotation of the domain models they use as inputs and outputs. The tools required for semantic annotation are not a software product of the current project but are provided by the caCORE SDK. It is the responsibility of developers to use the caCORE SDK tools to semantically annotate relevant domain models.

Installation of appropriate modified Bioconductor packages as caGrid analytic services involves steps identical to those required to install any collection of Java objects as analytic services, so the tools required for this step are not a software product of the current project. Steps specific to installing Bioconductor packages as web services will be documented; scripts or other functionality to automate specific steps in this process (e.g., coordinating installation of Bioconductor packages, their Java representation, and semantic annotation information) will be developed.

Invocation of Bioconductor functionality as an analytic service requires correct installation of R, Bioconductor, and SJava, in addition to standard requirements for caGrid analytic services. Our project will document recommended ways of installing R, Bioconductor, and SJava, and will provide installation software to automate this process, as appropriate.

2.3 USER CHARACTERISTICS

Primary users of this software include Bioconductor package developers and caGrid node administrators. A secondary user is the individual invoking Bioconductor functionality as an analytic service.

The Bioconductor package developer is a computationally and statistically capable individual wishing to produce or modify an R package to be deployed as an analytic service in caGrid. The individual is familiar with R, but has no special skills related to analytic services. The individual needs to be informed of the steps required for function and data type specification, and for producing Java classes corresponding to their Bioconductor package.

The caGrid node administrator is familiar with caGrid architecture and semantic annotation requirements of caCORE. The role of the administrator (defined more completely in the Bioconductor / caBIG™ use case document[2]) is to facilitate semantic annotation, and to install and deploy Bioconductor packages as web services. This individual may have limited familiarity with R, requiring that installation steps unique to R be documented and appropriate tooling (again, for the steps unique to this project) supplied.

2.4 CONSTRAINTS

a. Architecture. Our project is directed toward a current Linux operating system with standard development tools. These tools include those[3] required to build R. The system has versions of Java required for SJava[4] and caBIG™ [5]. Deployment and invocation tools require the web service infrastructure specified by caGrid[6]. Extension to use on Windows requires installation of specific but readily available software tools for R development, as indicated in footnote 3; Java and caGrid requirements on this platform are detailed in the documents cited earlier.

b. Data standards. Bioconductor packages analyze a diversity of data types; ensuring that these correspond to existing data types accessible as caGrid data services is the responsibility of individual package developers.

   Analytic services frequently have ‘tuning’ and other parameters required for function evaluation but largely irrelevant to semantic interoperability. Tuning parameters will be represented as semantically annotated objects, as required by caBIO. Large analytic services may have many parameters. Each CDEs will be created or reused for each

---

[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://cran.r-project.org/doc/manuals/R-admin.html

[4] http://www.omegahat.org/RSJava/

[5] https://cabig.nci.nih.gov/

[6] https://cabig.nci.nih.gov/workspaces/Architecture/caGrid

parameter. In addition, logical groups of parameters will themselves have CDEs created or reused. This follows the caBIG recommendations[7].

c. Programming language versions. R packages developed during this project will require the R-2.4.0. Requirements on Java are those specified by caGrid.

2.5 QUALIFICATION PROVISIONS

a. Specific code-based test facilities will be constructed to qualify requirements 3.1.*, 3.2.*, and 3.3.* (* represents all subsections).

2.6 ASSUMPTIONS AND DEPENDENCIES

This project relies on services provided by caDSR (to annotate and curate data types) and caGrid (analytic service infrastructure). Services used directly or indirectly include modules for semantic annotation (Introduce) and data registration (GME) during service creation, and Dorian for security verification both during service creation and invocation.

3. REQUIREMENTS

There are three requirement types indicated below. Requirements for caBIG™ are presented first. The RWebServices component is a software unit responsible for particular functionality. 'Exemplars' are specific Bioconductor functionality to be exposed as proof-of-methodology.

3.1 CABIG™ REQUIREMENTS

This section describes the requirements for this release in order to achieve caBIG™ Silver compatibility.

3.1.1 Vocabularies and data elements

One of the main goals of caBIG™ is to ensure that all applications use the same vocabulary and data elements for similar concepts across the grid (partial and full CDE reuse). This is aimed at eliminating inconsistency in data descriptors (metadata) and hence makes it possible to pass semantically equivalent data with confidence across various applications within the grid.

Req ID	Requirement Type	Requirement Description	CCR # (Optional)	Trace from User Requirement/ Trace to System Requirement (Optional)
3.1.1.1	caDSR Usage	Bioconductor / caBIG™ must use the Cancer Data Standards Repository (caDSR) and Enterprise Vocabulary Services (EVS) for each data element and concept that will be exposed to the grid.		

[7] slide 6, "Parameter Values in caBIG", George Komatsoulis, June 14, 2006 ICR teleconference.

3.1.2 User Interface (UI) Guidelines

As an analytic web service, Bioconductor / caBIG™ does not have a graphical UI.

3.1.3 API Access

The main goal of caBIG™ is to create a network or grid of applications, tools, and data, thereby creating a *World Wide Web* of cancer research. In order to achieve this, all silver-compliant caBIG™ applications must provide an Application Program Interface (API) so that others can retrieve meaningful data, correlate data with other applications, or create new applications.

Req ID	Requirement Type	Requirement Description	CCR # (Optional)	Trace from User Requirement/ Trace to System Requirement (Optional)
3.1.3.1	API Access	Bioconductor / caBIG™ must provide caCORE like APIs to expose functionality as an analytic service.		

3.1.4 caGrid ACCESS

All gold-compliant caBIG™ applications must act as data and/or analytical services on the caGrid so that others can retrieve and/or analyze meaningful data, correlate data with other applications, or create new applications.

Bioconductor / caBIG™ is funded for silver compatibility, and does not have requirements for caGrid access. Nonetheless, our intention is to provide caGrid enabled analytic services.

Req ID	Requirement Type	Requirement Description	CCR # (Optional)	Trace from User Requirement/ Trace to System Requirement (Optional)
3.1.4.1	caGrid Access	Bioconductor / caBIG™ must be caGrid enabled and act as a data and/or analytical service on caGrid.		
3.1.4.2	caGRID Access	All schemas of published services will be made available via the GME		

3.1.5 Environment

All caBIG™ projects should be platform- and database-independent applications. This means that it should be possible to run Bioconductor / caBIG™ on any operating system. Considering the large number of combinations of test scenarios, however, Bioconductor / caBIG™ must be certified for the environments listed in Table 1.

Table 1: System configurations for Bioconductor / caBIG™

Server operating system	Suse 10.0

Apache, Tomcat, Globus, Java, Ant	As specified by caGrid 1.0
R	2.4.0
SJava	0.6-4

3.2 R AS ANALYTIC SERVICES (RWEBSERVICES)

RWebServices is required to construct Java classes corresponding to S4 or core R data types and Java function signatures based on R functions annotated by TypeInfo. RWebServices constructs methods for dispatch of analytic services from Java to R. RWebServices augments SJava native translation between Java and R.

Req ID	Requirement Type	Requirement Description	CCR # (Optional)	Trace from User Requirement/ Trace to System Requirement (Optional)
3.3.1	Component	Produce Java classes corresponding to S4 or core R data types		
3.3.2	Component	Produce Java classes encapsulating equivalent R function signature(s)		
3.3.3	Component	Provide methods to dispatch analytic services for R evaluation		
3.3.4	Component	Provide (in conjunction with SJava) data translation services between Java and R internal representations of core R data types.		
3.3.5	Component	Provide error reporting functionality associated with R function evaluation.		
3.3.6	Component	Correctly return R results as Java data objects to analytic service requests.		

3.3 EXEMPLAR FUNCTIONALITY

Our tools enable annotation and publication of Bioconductor packages as analytic services, rather than wholesale conversion of Bioconductor packages for this purpose. Nonetheless, our project includes exemplars chosen from three different areas of Bioconductor functionality. The goal is to expose at least one function from each package as a caGrid analytic service.

3.3.1 Affy

The package contains functions for exploratory oligonucleotide array analysis. Functionality includes background correction, normalization, and expression summaries. Inputs are typically collections of microarray expression values representing data collected from several samples in a single experiment. Outputs are typically transformed expression values or summary presentations. The MAGE-OM[8] is used where possible.

[8] http://www.mged.org/Workgroups/MAGE/mage-om.html

Req ID	Requirement Type	Requirement Description	CCR # (Optional)	Trace from User Requirement/ Trace to System Requirement (Optional)
3.3.1.1	Exemplar	Expose specific functionality as a web service, using TypeInfo, RWebServices, and SJava. Exposed data elements are associated CDEs.		
3.3.1.2	Exemplar	Receive MAGE-OM object		
3.3.1.3	Exemplar	Invoke and perform analytic service corresponding to the 'expresso' function in the 'affy' package		
3.3.1.4	Exemplar	Add resulting transformation and documentation to MAGE-OM object		
3.3.1.5	Exemplar	Return transformed MAGE-OM object from web service.		
3.3.1.6	Exemplar	Report errors during execution		
3.3.1.7	Exemplar	Data system overview (Powerpoint or equivalent)		
3.3.1.8	Exemplar	UML model of data system (Enterprise Architect)		
3.3.1.9	Exemplar	Semantically annotated XMI files		
3.3.1.10	Exemplar	SIW report		
3.3.1.11	Exemplar	UML loader checklist		
3.3.1.12	Exemplar	Value domain report		
3.3.1.13	Exemplar	Vocabulary report		
3.3.1.14	Exemplar	Standrds report		
3.3.1.15	Exemplar	Full CDE use report		
3.3.1.16	Exemplar	API documentation		

3.3.2 PROcess

PROcess is a package for processing protein mass spectrometry data. Functionality includes baseline subtraction, peak identification, and quality control. Inputs are typically collections of SELDI-TOF mass spec scans representing data from a single experiment. Outputs are typically transformed scans, peak identification vectors, or summary presentations. The mzXML domain model[9] is used where possible (for input parameters)

Req ID	Requirement Type	Requirement Description	CCR # (Optional)	Trace from User Requirement/ Trace to System Requirement (Optional)
3.3.2.1	Exemplar	Expose specific functionality as a web service, using TypeInfo, RWebServices, and SJava. Exposed data elements with associated CDEs.		
3.3.2.2	Exemplar	Receive mzXML object		
3.3.2.3	Exemplar	Invoke and perform analytic service, reporting correct result to user		
3.3.2.4	Exemplar	Return aligned peaks in a CDE-defined data structure.		

[9] http://tools.proteomecenter.org/mzXMLschema.php

3.3.2.5	Exemplar	Report errors during execution		
3.3.2.6	Exemplar	Data system overview (Powerpoint or equivalent)		
3.3.2.7	Exemplar	UML model of data system (Enterprise Architect)		
3.3.2.8	Exemplar	Semantically annotated XMI files		
3.3.2.9	Exemplar	SIW report		
3.3.2.10	Exemplar	UML loader checklist		
3.3.2.13	Exemplar	Value domain report		
3.3.2.12	Exemplar	Vocabulary report		
3.3.2.13	Exemplar	Standrds report		
3.3.2.14	Exemplar	Full CDE use report		
3.3.2.15	Exemplar	API documentation		

3.3.3 DNAcopy

DNAcopy segments DNA copy number data collected from array CGH experiments, using circular binary segmentation to detect regions with abnormal copy number. Functionality includes smoothing and segmenting copy number data. Typical inputs are vectors corresponding to marker location and sample copy numbers. Outputs are vectors of sample identifiers, segment start and stop positions, and related statistics.

Req ID	Requirement Type	Requirement Description	CCR # (Optional)	Trace from User Requirement/ Trace to System Requirement (Optional)
3.3.3.1	Exemplar	Expose specific functionality as a web service, using TypeInfo, RWebServices, and SJava. Exposed data elements with associated CDEs.		
3.3.2.2	Exemplar	Receive CDE-defined data object representing raw DNA copy number data.		
3.3.2.3	Exemplar	Invoke and perform analytic service, reporting correct result to user		
3.3.2.4	Exemplar	Return CDE-defined data object describing marker location and copy number.		
3.3.2.5	Exemplar	Report errors during execution		
3.3.2.6	Exemplar	Data system overview (Powerpoint or equivalent)		
3.3.2.7	Exemplar	UML model of data system (Enterprise Architect)		
3.3.2.8	Exemplar	Semantically annotated XMI files		
3.3.2.9	Exemplar	SIW report		
3.3.2.10	Exemplar	UML loader checklist		
3.3.2.13	Exemplar	Value domain report		
3.3.2.12	Exemplar	Vocabulary report		
3.3.2.13	Exemplar	Standrds report		
3.3.2.14	Exemplar	Full CDE use report		
3.3.2.15	Exemplar	API documentation		

4. SYSTEM ARCHITECTURE

4.1 SYSTEM ARCHITECTURE OVERVIEW

The high-level concept of the system is the development and exposure of R functionality as web services for statistical analysis of high-throughput biological data. In general the statistical methodologies are previously defined, and their specification is outside the scope of this document. The focus here is on two main aspects of the project: service creation from existing analytic functionality, and service evaluation in the caGrid services environment.

Statistical analytic routines are existing functions written in the R programming language and distributed as R libraries. Extensive documentation of the R programming language is available[10], as are details of the analytic functionality available in existing R / Bioconductor packages[11].

The system architecture for service creation consists of an existing Bioconductor package (e.g., *affy*, *PROcess*, or *DNAcopy*, above) coupled with the R libraries TypeInfo (previously developed) and RWebServices (to be developed here). The architecture relies on user access to simple text file creation utilities, a command line environment, and system software requirements for the R user environment. In general, R runs on modern Linux, MacOS, and Windows desktop or server computers with standard software tools.

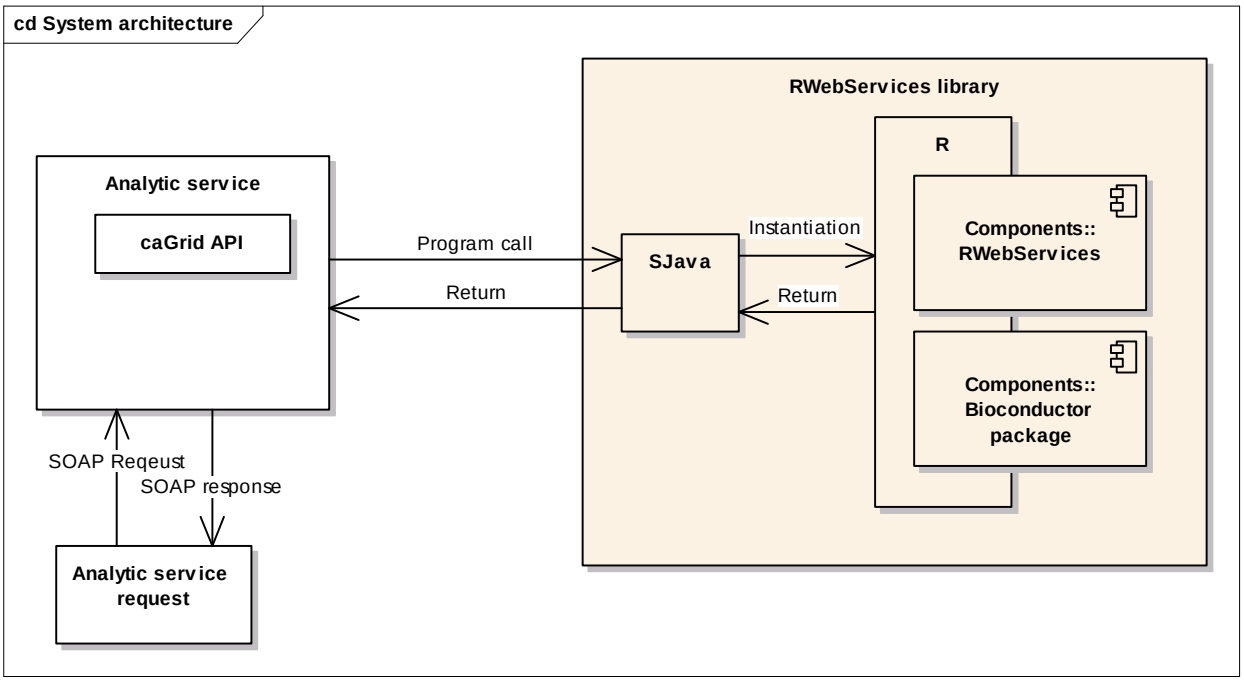

Figure 1 System architecture block diagram – service invocation. Tan objects represent components provided by Bioconductor / caBIG™.

[10] http://cran.fhcrc.org

[11] http://bioconductor.org

The system architecture for service invocation consists of an established caGrid node with installed Java. Software components provided by this project include SJava, R, Bioconductor package(s), and the RWebServices package.

Safety, security, and privacy design decisions are under the purview of caGrid services.

4.2 ARCHITECTURAL DESIGN

For analytic service creation, the basic control and data flow starts with an existing Bioconductor package and a TypeInfo annotation file identifying functionality, argument, and return types to be exposed from the package as web services. The package and TypeInfo annotation file are processed by TypeInfo and RWebServices to create a hierarchy of Java data class objects with data element accessor methods, and a Java class with Bioconductor functionality exposed as strongly typed Java functions. An additional Java class coordinates the requests for analytic services with the R evaluator. The Java data elements and functionality are semantically annotated using caCORE SDK. The semantically annotated functionality is exposed as a grid analytic service using the caGrid SDK tools. This work flow is documented in detail in the Bioconductor Use Case document[12]; the architectural design of the caCORE and caGrid SDKs are outside the scope of this document.

The remaining paragraphs in this section refer to software units participating in analytic service invocation, and diagramed in Figure 1.

The component labeled 'web service' represents caBIG™ infrastructure outside the scope of this document. In the context of the present project, the resource use of this infrastructure is likely to involve substantial data transfer bandwidth (data files of 10's-100's of MB) and corresponding memory.

R and Bioconductor are existing software packages for statistical computation. Extensive documentation of the R programming language is available[13], as are details of the analytic functionality available in existing R / Bioconductor packages[14].

SJava is an existing software package that embeds R in a Java environment. The purpose of SJava in our implementation is as an interface between caGrid and R. The requirement placed on SJava for the present project is native translation between web services and R representations of data objects, and the ability to invoke R functions from Java. The current version of SJava is 0.69-4. Documentation for SJava is available[15].

---

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

[13] http://cran.fhcrc.org

[14] http://bioconductor.org

[15] http://www.omegahat.org/RSJava/

RWebServices is a component of R. Its function in the context of service invocation is to correctly construct Java methods for dispatch of analytic service requests to R functions, and to extend native SJava data translation between Java and R. RWebServices is currently in development. It has minimal resource use requirements beyond those of R.

Exemplar packages are components of R. Packages provide analytic service evaluation, as detailed in section 3.3. Analytic components of packages are fully developed. Interfaces consistent with caBIG™ requirements are under development. Resource use is highly variable depending on service request. Typical memory requirements are 1-2GB with computation times of 1-2 minutes on systems outlined above. Very large service requests may impose significant memory requirements requiring operating systems able to address >4 GB of memory. These requirements are imposed by the underlying R software and algorithms to be exposed (recall, Bioconductor / caBIG is not developing new algorithms), and must therefore be address by deployment on sufficiently enabled hardware systems. Large data objects also impose significant communication costs; effectively communicating such objects is outside the scope of Bioconductor / caBIG™.

4.3 INTERFACES

The primary interface to Bioconductor / caBIG™ is as caGrid service.

- External Software Interfaces – Bioconductor / caBIG™ interfaces with caGrid as an analytic service. The interface requirements are as specified by caGrid.
- Software System Interfaces – There are no special system software interfaces required for Bioconductor / caBIG™.
- Software System Components Interfaces – Interfaces between RWebServices and SJava are outlined below; there are no additional system software component interfaces.
- Hardware Interfaces – There are no hardware interfaces beyond those required by caGrid.

4.4 SYSTEM SECURITY

- Network Level - Bioconductor / caBIG™ relies on network security provided by the caGrid infrastructure.
- Database Level – There are no database-level system security issues.
- Application Level –Security within R is problematic, because an experienced could gain full access to the R programming language. Our security model is to limit opportunities for harmful consequences through separately invoking R processes for each service request, to avoid data caching between sessions, and to run the R process with minimal local credentials.

5. COMPONENTS

5.1 RWEBSERVICES

Note: Please do not confuse the name 'RWebServices' with functionality. This component includes provision of grid services as outlined below and in section 3.2. Changing the name of the component to more accurately reflect its role is an in-process discussion.

5.1.1 Type

R package.

5.1.2 Purpose

The purpose of this component is to provide a Java interface to existing Bioconductor functions, as specified in requirement section 3.2.

5.1.3 Function

The component has two distinct functionalities.

During analytic service creation, RWebServices transforms a list of TypeInfo-annotated functions to a hierarchy of Java data classes (corresponding to the R S4 types indicated by TypeInfo) and methods for evaluating functions.

During analytic service invocation, classes created by RWebServices dispatch analytic service requests for evaluation in R. RWebServices also provides native bidirectional translation between Java objects and their corresponding R S4 classes.

5.1.4 Subordinate

None.

5.1.5 Dependencies

RWebServices depends on TypeInfo during analytic service creation, and SJava during service invocation.

5.1.6 Preconditions

RWebServices requires existing Bioconductor packages with functions containing TypeInfo attributes. During invocation as an analytic service RWebServices requires caGrid infrastructure to correctly specify data elements and functions and SJava methods for R evaluation.

5.1.7 Interfaces

5.1.7.1 Analytic service creation

RWebServices takes as input a list of TypeInfo-annotated functions. It produces as output a hierarchically-arranged collection of Java data and function classes. Each Java data class contains private members corresponding to the data elements of the corresponding S4 object. There are public methods to 'get' and 'set' each data element. The hierarchy localizes data definitions to folders specific to the R library in which the data or function was specified, e.g., bioc.affy.function.expresso.

RWebServices has access to the R libraries where the function is originally defined, and where data objects referenced by the type information attributes are defined.

5.1.7.2 Analytic service invocation

Classes created by RWebServices provide an interface between analytic services and R function invocation. The interface exposed as analytic service consists of a function name and strongly-typed argument and return values. The interface is augmented by semantic annotation services provided by the caCORE SDK.

The interface to RWebServices during analytic service invocation also follows the requirements and specifications of SJava. Specifically, RWebServices contains C-level converter functions that augment SJava functionality. The C-level converter functions accept and return the R SEXP data structure. Translation services provided by RWebServices in both directions is performed on the R, rather than Java, side of the bridge.

5.1.8 Resources

Not applicable.

5.1.9 References

Online and package documentation of the SJava interface is available[16].

5.1.10 Processing

5.1.10.1 Analytic service creation

R pseudo-code for the overall functionality of analytic service creation is as follows:

```
RJavaSignature <- function(inputs) {
    typeInfo <- typeInfo2Java(inputs) # extract type information
    pkdList <- pkgList(typeInfo) # identify R packages
```

[16] http://www.omegahat.org/RSJava/

```
    jTypeInfoByPkg <- typeInfoByPkg(inputs, pkgList) # data:pkg & function:pkg map
    buildJavaDataClasses(jTypeInfoByPkg) # build data bean classes
    buildRWebServicesInterface(jTypeInfoByPkg) # build function interface classes
}
```

Data class creation pseudo-code is:

```
buildJavaDataClasses <- function(jTypeInfoByPkg) {
    pkgTypes <- typesByPkg(jTypeInfoByPkg) # classify structure – primary, S4, ...
    for (pkg in pkgTypes) { # each package separately
       for (type in pkg) {
           # create Java equivalent of new types
           # createXXX constructs classes and writes to disk
           if (type == "primary" || type== "S3") next # no action needed
           else if (type == "S4') createS4Bean(type)
           else if (type == "ClassUnion") createClassUnionBean(type)
           # handle additional / future types
           else createUnknownBean(type) # map to RUnknown
       }
    }
}
```

Function creation pseudo-code is:

```
buildRWebServicesInterface <- function(jTypeInfoByPkg) {
    rGlobalClass <- createOrGetGlobalEnv() # global wrapper for R functions
    for (pkg in jTypeInfoByPkg) { # each package separately
       jClass <- createRWebServicesInterfaceTemplate() # template
       for (funcs in pkg) {
           addInterfaceMethod(jClass, funcs) # interface to R wrappers
           addWrapper(rGlobalClass, funcs) # wrappers to R
       }
       writeJClass(pkg, jClass) # save to disk
    }
    writeRGlobalClass(rGlobalClass) # save / update global wrapper
}
```

5.1.10.2 Analytic service invocation

Service requests arrive and are decoded by caGrid functionality. The service request is then dealt with by the following pseudo-code:

```
rws = RWebServicesServ(); // Start R evaluator, load R libraries and converters, etc
try {
       result = rws.analyticService(serviceArgs); // invoke service with args
} catch (java.rmi.RemoteException e) {
       reportException(e);
}
return result;
// rws terminates R evaluator when out of scope
```

The `.analyticService` invokes a method in `rGlobalClass` to call and evaluate, through SJava, the R function. The returned result is packaged and forwarded to the requesting service by caGrid.

Program flow for each data translation function is straight-forward: create a new instance of the target object corresponding to the source object, populate the target object with source data, and return the target object. Populating the target object with data typically involves a memory copy.

There are no provisions for user interrupts; this is not a usual part of the web services workflow.

5.1.11 Data

There are three types of data internal to this component. R objects are represented as Java data classes, typically with a structure:

```
package biocJavaMap.rGlobalEnv.data;

public class RClass extends rtypes.RObject {
       private rtypes.RNumeric eps; // data slot
        public RClass () {} // creation
       public void setEps(rtypes.RNumeric eps) {…} // set
       public rtypes.RNumeric getEps() {…} // get
       public String toString() {…} // string representation, for debugging / testing
}
```

Java function interfaces are structured as

```
package biocJavaMap.RWebServicesServ;
import org.omegahat.R.Java.ROmegahatInterpreter;
import org.omegahat.R.Java.REvaluator;
import java.util.*;

public class RWebServicesServ {
       private REvaluator e; // local evaluator
       private biocJavaMap.rGlobalEnv.function.rGlobalEnv myrGlobalEnv;

       public RWebServicesServ() throws Exception {…} // creation
       public rtypes.RNumeric rGlobalEnv_RFunc1(funcArgs1) {…} // invocation
       public rtypes.RNumeric rGlobalEnv_RFunc2(funcArgs2) {…} // invocation
       …
}
```

The R function wrappers are in a `rGlobalEnv` data structure:

```
package biocJavaMap.rGlobalEnv.function;
import org.omegahat.R.Java.REvaluator;
import java.util.*;

public class rGlobalEnv {
       // initialize R evaluator with R function defintions and data converters
       public REvaluator initializeEvaluator(REvaluator e) throws Exception {…}
       // methods for each defined R function; calls R through REvaluator
       public rtypes.RNumeric rFunc1(funcArgs1) throws Exception {…}
}
```

SJava defines the `REvaluator` object.

5.1.12 Space estimates

Space estimates for internal data objects are minimal. Run-time space estimates for data objects can be large (1-2GB), depending on data sizes of service requests.

5.1.13 Impact to existing components

Not applicable

5.1.14 System security

System level security information is outlined in section 4.4.

6. SIGN OFF

6.1 APPROVAL

Architecture Workspace Rep	***Print Name***	***Date***
VCDE Workspace Rep	***Print Name***	***Date***
Adopter Rep	***Print Name***	***Date***

APPENDIX A – ACRONYM LIST

Term/ Abbreviation	Description
DBA	Database Administrator
PM	Project Manager
RM	Requirements Manager
RTM	Requirements Traceability Matrix
SCM	Software Configuration Management
SDLC	Software Development Lifecycle
SDP	Software Development Plan
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
TBD	To Be Determined
TM	Test Manager