Building a Biodiversity Problem-Solving Environment

Richard White[1], Andrew Jones[1], Nick Pittas[1], Alex Gray[1], Xuebiao Xu[1], Tim Sutton[2], Oliver Bromley[3], Neil Caithness[2], Nick Fiddian[1], Alastair Culham[2], Frank Bisby[2], Shonil Bhagwat[4], Peter Brewer[2], Chris Yesson[2], Paul Williams[4]

[1] School of Computer Science, Cardiff University, Cardiff CF24 3XF
[2] School of Plant Sciences, The University of Reading, Reading RG6 6AS
[3] School of Biological Sciences, University of Southampton, Southampton SO16 7PX
[4] Department of Entomology, The Natural History Museum, Cromwell Road, London SW7 5BD

Abstract

In the Biodiversity World project, our goal is to create a flexible and extensible Grid environment in which heterogeneous distributed biodiversity-related data sets and analytical tools are made available to be located and assembled by users unfamiliar with the Grid into workflows to perform complex scientific analyses. We have previously defined and described an architecture containing a number of kinds of components which conform to a common model of Grid interoperability. In this paper, we concentrate on the components and strategies which are important for the assembly of a complete working environment. We describe recent work on three main components of the environment: the workflow system, which provides both workflow design and enactment facilities based on Triana; the metadata repository, implemented using simple flexible entity-attribute-value triples; and the user interface, which consists of parts to control the workflow engine, the metadata repository and tasks as they are executed. We discuss various issues which arose during this work, some of which have not yet been fully resolved, including data set management and access, resource location and invocation, metadatabase representation, wrapping techniques for legacy resources, and user interaction with them.

1. Introduction

The Biodiversity World (BDWorld) project[1] is building a "problem-solving environment" to enable scientists to perform analysis procedures consisting of complex sequences of operations involving the acquisition of data sets and their analysis.

At previous All Hands meetings, we have described the BDWorld 'resources' [1], both data access systems and analytical tools, which provide 'operations' that can be invoked and which typically read existing data sets and deliver new data sets. They are supported by a toolkit to provide this functionality either directly or by means of wrappers for legacy resources. We also described the overall BDWorld architecture and, in particular, the BDWorld-Grid Interface (BGI) that provides Grid infrastructure independence for BDWorld components. We have demonstrated a proof-of-concept prototype system [2] using "hard-wired" workflows.

In this paper, we shall focus on the design decisions which support the system and the processes by which new resources are brought into the environment and made interoperable.

1.1 Research scenarios

To assist in the design and implementation of a system which is intended for use by scientists who are not specialists in computing or the use of the Grid, we chose one general area of application in which we have experience, biodiversity informatics, and three example study areas or 'scenarios', each addressed by a small team of scientists with specific research objectives.

These three scenarios have been previously described [2]. They comprise the quantification of the variation of species diversity with geography, the modelling of the geographical distribution of individual species as explained by climate variables in order to predict past and future climate-related changes in distribution, and the enhancement of phylogenetic inference by adding information about the geographical distribution of species to the usual morphological and molecular data characterising the species.

This set was chosen so that some of the resources to be integrated into the environment would be used in more than one scenario, to

[1] www.bdworld.org

provide a richer set of resources available in each scenario and reduce the effort of integrating them.

2. Design goals and principles

We began with the premise that, within these three example scenarios, there are a number of data and analysis resources which would form the basic set of tools that the scientists would wish to access. The first issue to be resolved was the question of how users would wish to interact with them.

2.1 Styles of user interaction

In the BDWorld project we aim to provide an environment in which biologists can readily perform a wide range of biodiversity-related tasks. Some of these tasks are complex but repetitive in nature, and for this it is desirable to have a repository of pre-constructed workflows that can be re-run and modified as appropriate. On the other hand, the ability to explore and analyse the data sets available in an incremental manner, and to build workflows from the fragmentary workflows that are thereby in effect designed, is also important. Otherwise scientists are constrained to specify tasks in a way that may be quite different from the way they may prefer to work.

We have commenced by considering the use of workflows in BDWorld, and our primary objective for user interaction is the provision of support for the kinds of user interface required for workflow-oriented interaction, rather than the more exploratory scenario identified above.

2.2 The problem-solving environment

The decision to produce an environment in which the construction and later execution of a workflow is the central activity has clear implications for what the main components of the BDWorld system might be.

In addition to the resources which will provide the tasks to be executed, the system requires a workflow designer, an enactment engine to execute the workflow, a metadata repository to describe resources and user interfaces to all these components.

2.3 Grid interface

Although the overall aim of the project is to demonstrate one approach to the application of the Grid to scientific research, the rapid change and refinement in Grid protocols, and the competing Grid software currently available, makes it desirable that the environment we build is not bound inextricably to a particular point in the evolution of the Grid, which would render the environment almost instantly out of date. Instead, we attempt to insulate the environment from changes in the Grid by insisting that all Grid communications between the main software components are channelled through a communications layer (BGI) which isolates the evolving Grid technologies from the resources behind an application programming interface (API).

This has an additional benefit to the project, that only one of the programmers, who implements and maintains the BGI, needs to have detailed knowledge of Grid protocols and software. The other programmers only require knowledge of the API.

3. Recent developments

We now have a working BDWorld system in which all the key components are in place, including flexible user-defined workflows, although some features need to be enhanced. Work is currently in progress to enrich the BDWorld environment by adding new resource components as they are required for use in the three areas of active biodiversity research being used as test scenarios in the project.

A number of workflows have been specified on paper for these scenarios, and a series of prototypes has been implemented in which some of these workflows have been “hard wired”. The most recent prototype provided a Web interface to a system in which the workflow enactment was provided by a simplified, proprietary enactment engine. Although the workflows could be modified, no tools were provided for doing this and so they had to be edited by hand.

In this paper we describe the key features of the various classes of components, especially those which have not previously been described in detail. The main features and components of BDWorld on which we have been working recently are described in the following sections.

3.1 Workflow system

A workflow design environment will allow scientific users to develop applications which invoke sequences of tasks. It will provide a user interface for the scientist to control task assembly and execution.

We investigated and compared various workflow systems, such as IBM’s WSFL/ BPEL4WS, MQSeries Workflow and Flowmark; Microsoft’s MSCMS 2001 Web Author; HP’s OpenPM and OpenFlow; EBI’s Taverna and Talisman; Sun and BEA’s WSCI;

WfMC's XML supported products such as OBE; Praxis's FlowMake; ebBPSS; W3C's XML Pipeline; BPEL; OFBiz; OpenWFE and wftk; StateFrame from Alia Systems; TENT, Triana and JXTA; Apache's Struts; Breeze and WfXML.

We decided to adopt the Triana workflow system[2] [3]. Our reasons include Triana's advantages for easy workflow assembly, window-based visualisation tools for both designing and executing workflows, its two-concept simplicity while keeping the effectiveness of the stand-alone workflow enactment system, embedded features provided to facilitate the authoring of tasks and units, its capability for distribution and portability among different systems using Java and language independence with OCL, and its extended Web Services/P2P features. But most important of all, Triana has been developed at Cardiff and we have direct access to the Triana team for support.

We are enhancing and adapting Triana to provide both a workflow designer and a workflow enactment engine that can interface to BDWorld. It has been adapted to facilitate its use by scientific users to design and assemble workflows composed of tasks selected from the operations provided by the resources.

These tasks have to be predefined to comply with the required specification for Triana tasks, units and types. A template is used for designing working Triana units to correspond with the underlying resources. Since there is no direct one-to-one mapping relationship between the existing BDWorld resources, operations and data set types and their properties and the corresponding Triana task, data type and unit concepts, a customisable conversion component is necessary in order to implement the bi-directional mapping between the BDWorld resources and the operations they implement and the corresponding Triana groups and unit grouping.

We are currently developing the first of a series of BDW prototypes that use Triana for the construction and enactment of workflows. However, although Triana has proved effective if considered as a visual programming tool, in its present form it has a number of limitations as a basis for a Problem-Solving Environment, and in future prototypes we shall be adapting and enhancing Triana to address these issues. The limitations, and our plans for dealing with these limitations, include the following:

- Triana was originally designed for sequencing tasks relating to astrophysics, in which large data volumes, but a relatively limited range of data types, must be accommodated. The need in Triana to define one's own Triana data types corresponding to each of those used in the remainder of the system is a tedious process, even for a small number of types. But BDWorld must be able to manage a large, and growing, range of data types. Our approach to dealing with this problem is that, during enactment, BDWorld data remains encapsulated as RemoteData objects, except where user interaction is required.
- Triana supports resource discovery by means of a hierarchical tree menu. This is quite limiting as users may wish to specify their criteria for resource discovery in a range of different ways. The knowledge against which users' criteria must be matched is held in a metadata repository (described below); we therefore require a user interface which allows users to express constraints on resources required and which controls a new Resource Locator module which interacts with the metadata repository to satisfy such queries.
- Triana allows units with compatible interfaces to be connected and manipulated in a workflow if they are syntactically compatible, which requires that the data types of data items to be read by a target unit or task are the same as the data types of the data to be written by the source unit or task. However, it may be difficult to know whether they are also semantically compatible. It is not yet clear how to introduce the necessary metadata and constraint information into Triana in a simple but effective way. This is an example of the more general problem of determining and comparing the behaviours and competencies of interacting tasks/units – we plan to cope with this issue with some ontology-based mediation technology.
- Extensive provenance data associated with workflows is required in BDWorld. For example, the versions of data sets and analytic tools used must be recorded, and storage of intermediate results is often useful: someone modifying a workflow will then not need to re-execute it from the start, but only from where it has been changed.
- In modifying a workflow, a user may very well want to know about what alternatives there are to a particular element. A context-sensitive user interface that can provide such information is therefore an important requirement.
- At present, Triana's usual user interface is not Web-based, and our requirements include the provision of a Web portal for transient

[2] www.triana.co.uk

users. A web interface to Triana is currently under development by the Triana team, and we intend to make use of this in due course, adapting it as necessary.

3.2 Metadata repository

Metadata about resources and the operations that they support is being stored in a repository. The metadata repository (MDR) is implemented as a specialised BDWorld resource, avoiding the need to define a separate protocol for its use.

Resources invoke MDR operations indirectly via the BGI registerResource method to register their metadata, and the repository in turn answers requests to supply information to the workflow system, to assist the user to locate suitable tasks, ensure their compatibility and manage any intermediate data sets. For example, the workflow system can interrogate the MDR to retrieve suitable resource operations using the MDR's getOperations operation.

Prior to use, a BDWorld resource registers itself using registerResource, using XML to express its mandatory and optional resource metadata. Each concept which has metadata is represented as an 'entity' which may be described by several 'attributes', each of which has a value. Every item of metadata is thus represented in the repository as a triple (entity, attribute, value). The entity element identifies the thing or concept being described, the attribute is the parameter in question and the value is the parameter's current state.

Such triples have been shown to be able to represent a variety of different formats of data and data structures. Algorithms exist to transform predicates of any entity into a set of binary predicates, so the stipulation that we use triples only is not significantly restrictive [4]. The value may be text, or may name another entity in the metadatabase or may refer to an external location such as a URI. It may also refer to a file such as a Word document or an XML schema, so that a collection of ancillary information can be stored in a structured form, associated with each resource, and be available for downloading, although it may not form part of the base metadata.

The triples are stored in a single simple MySQL database table with three columns. The main software component of the repository communicates with other components and resources through the BGI, and implements operations which store and retrieve metadata.

Every entity has the attribute "is a" to specify what kind of entity it is. Its other attributes vary accordingly. For example, the attributes which can be used to describe a resource include:

name	A readable short name for the resource
category	Broad category of the resource e.g. "locality"
uri	The location of the wrapper
operation	The name of an operation which can be invoked

As the project proceeds, these attributes will be refined, so that information registered by the resource keeps pace with the understanding of the needs of the workflow manager and the user.

An example fragment of the metadata for a resource which provides operations to perform tasks in the species distribution modelling scenario follows:

entity	*attribute*	*value*
CSM	is a	resource
CSM	name	CsmWrapper
CSM	category	species distribution model
CSM	uri	http://csm.rdg.ac.uk/csm/
CSM	operation	load layer set
CSM	operation	point converter

The various alternative options to represent information about workflows and the results of their execution are currently being considered in relation to the requirements of Triana and the ways in which we are extending it with a Resource Locator module to support resource discovery and a Resource Matcher to perform compatibility checking.

For example, at present we are testing the system with simple identifiers for each type of data set – if one operation writes a data set with the same identifier as the input data set required by another operation, then the operations are considered to be compatible and can be used as tasks in the same workflow.

Later, we will gradually enhance the metadata describing these data sets to support more sophisticated semantic reasoning about them, such as testing the compatibility of data matrices using their size information. Also, the addition of more terms which a user is likely to search for, in the manner of a thesaurus or ontology, will allow reasoning with synonyms, for example.

Similarly, more detailed knowledge of data formats will permit the user or the workflow designer to search for a suitable conversion tool or control a flexible format converter. For example, a simple XML format is used internally for locality data, into which all locality query results are converted; this process can be automated and controlled by metadata.

3.3 User interfaces

The user interface to the BDWorld environment consists of several parts, which provide user access to the workflow designer and enactment engine, including interaction with the individual tasks being executed, and to the metadata repository.

The main user interface which the scientists will use to interact with the Triana system is being enhanced to allow them to manage the workflow system, control task assembly and execution of a workflow, and control the tasks and data sets before, during and after workflow execution within a visual environment.

Triana assumes that interaction with the user will generally be localised and straightforward, for example by the use of dialogue boxes. In contrast, in BDWorld, complex interactions may be required during execution, e.g. selection of regions on a map, and some of these are most naturally done by direct communication with a remote resource. Thus various new components will need to be developed as adjuncts to Triana, supporting various kinds of interaction including interaction with remote resources via the BGI, Web-based dialogue or some internal triggered mechanisms, as discussed below in section 4.3. Also, if the system is to be extensible it is desirable for the introduction of a new resource to be supported by a mechanism for introduction of appropriate new interactive components that can be incorporated into the Triana-based system on demand.

A separate web interface to the metadata repository allows the metadata to be inspected and enhanced, as described above. This is intended mainly for systems developers and resource integrators. However, it also allows users to browse the MDR in order to explore the resources and data sets available and archived workflows, with the intention of re-using workflows or parts of them and the associated data that they generated. The web pages are generated on demand from templates, populated by JDBC from the data triples stored in the database. An index page lists the entities for which data is held. From this page it is possible to navigate to a detail page which lists the attribute values for a chosen entity. Other pages can be requested which display the metadata sorted by entity, by attribute and by value respectively.

Thus, although the main function of the MDR is to store BDWorld metadata, the flexibility of the structure and the standard way the data can be displayed in a browser means it has a variety of other applications both within the BDWorld project and elsewhere.

3.4 Integrating resources

We are adopting a flexible approach to the problem of integrating resources to work in the BDWorld Grid problem-solving environment. Figure 1 shows seven numbered columns representing different options for building or adapting a resource or other BDWorld component to work with BDWorld, depending on whether it is an existing ("legacy") or purpose-built resource, whether it is written in Java, and so on. Some BDWorld components, such as the MDR, are implemented in the same way as other resources, and will be considered together with the "true" resources which provide the operations which are available for use in workflows.

In all cases, "the Grid" (which may take various forms, as far as we are concerned) provides a foundation through which the resource or component communicates with other components of the BDWorld system. In most cases, as described above, direct communication with Grid protocols and software is provided through the BDWorld Communications Layer, which exposes an API, the BGI, which components use to communicate with other components.

The BGI defines the interface between the resource-facing side (Abstract Resource) and the Grid-facing side (BDWorld Communications Layer). An implementation of the latter for a particular Grid communications protocol will contain protocol-specific code to access the Grid and implement the BGI methods which are called by resources, such as registerResource. Conversely, resources or other components such as wrappers (see below) interface with the BGI by inheriting from the Abstract Resource class and providing implementations of the other BGI methods that it defines, such as invokeOperation, which the Communications Layer uses to access resources.

The Resource Base is implemented by a Java class which groups several commonly used functions which all resources and wrappers will need, such initialising their state at start up, negotiating authenticated proxies, preparing to register metadata with the MDR and handling the packing and unpacking of references to input and output data sets passed over the BGI when an operation is invoked. Each resource or wrapper inheriting from this Resource Base is then only expected to implement the invokeOperation method in order to meet the requirements of the BGI. The Resource Tools provide additional methods which are optional in the sense that they are useful only to particular types of resources and wrappers.

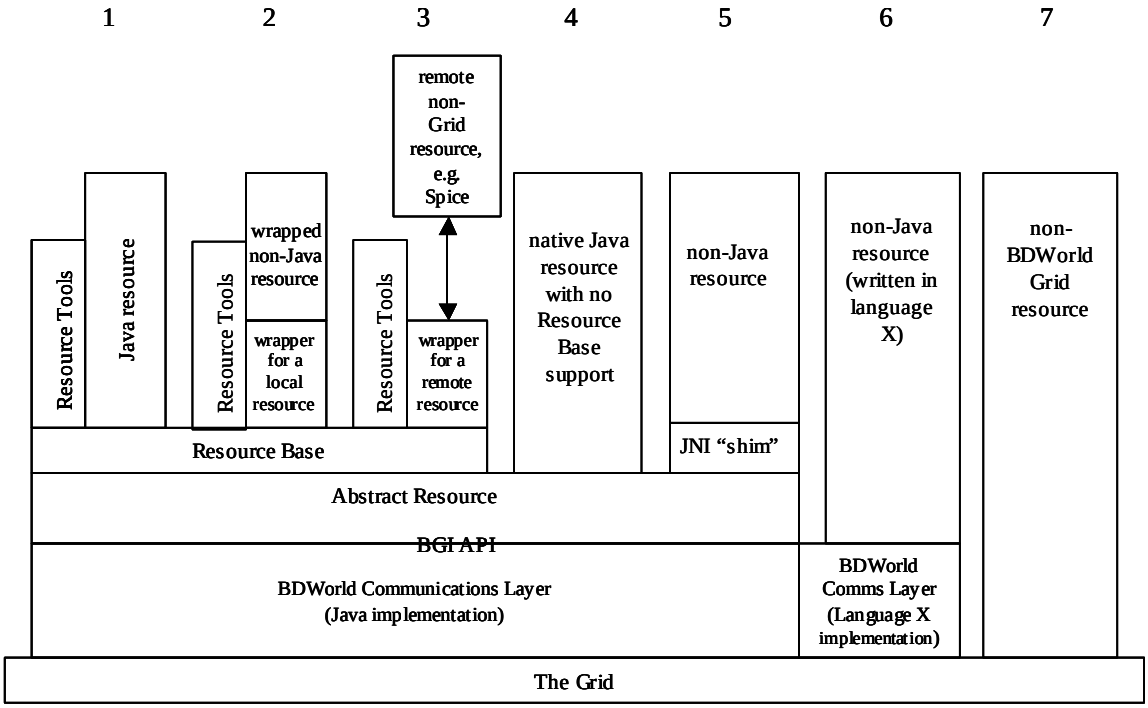

Figure 1: Options for BDWorld resource integration

Variations on this implementation strategy are illustrated in Figure 1. For example, column 1 represents a resource written in Java, typically specially for the BDWorld project (a Java "native" resource), such as the MDR. In column 2, a resource written in some other language than Java is accommodated, using a wrapper on the same machine to carry out all the BDWorld-related operations; the wrapper calls the non-Java resource, by an appropriate mechanism when one of its operations is invoked. A wrapper for local command-line programs would be an example.

Integration option 3 also uses a wrapper in a similar way, but the resource being executed is on a machine remote from the wrapper. The connection to the remote resource can be by any available means that the resource permits, and is not constrained by the Grid or other aspects of BDWorld. This allows for connecting to legacy data sources on the Internet, such as the Spice system run as a Web Service by the Species 2000 project to deliver species names and synonymy information.

The BDWorld Project Team is currently implementing resources using methods 1, 2 and 3. It is one of the design goals of the project that the BDWorld system should be available to be extended by other users and developers. The source code, including the Java classes mentioned above, will be released into the public domain to allow developers to use methods 1, 2 and 3. An alternative approach for other developers is to eschew our support classes and connect directly to our BGI, which is the technique shown in column 4, but inheriting from the Abstract Resource class would still be a sensible way to implement the API interface to the BGI.

There are also other possible approaches to the integration of non-Java resources, in addition to methods 2 and 3 already described, which have not yet been tried. Approach 5 is a variant of approach 4 which is made suitable for non-Java resources by the use of a thin Java "shim" which implements the Abstract Resource methods by using the JNI (Java Native Interface) package to call non-Java code.

Approach 6 would replace the Java communications layer which BDWorld uses to address the Grid by an equivalent layer written in some other language; this would then support resources written in that language, using an interface different from the BGI. Like the BDWorld Java communications layer, this layer would have to be re-written to accommodate changes in Grid protocols. However, because the BDWorld communications layer uses techniques such as XML data representation when communicating with the Grid, which are independent of the implementation language, such an alternative communications layer would still allow resources based on it to interoperate fully with BDWorld resources.

In contrast, approach 7 describes resources which communicate with the Grid without using any BDWorld protocols, such as MyGrid[3] [5] resources for example, and interoperability with such resources has not yet been investigated. They could of course be wrapped using one of the other approaches described above.

One strategy to adapt certain resources to work in the BDWorld environment is to create a so-called "generic" wrapper. A standard interface or API is defined for communicating between the wrapper and the resource, so that one wrapper can serve a number of similar resources. In the case of a Matlab wrapper currently implemented, this interface is a command-line syntax for a single operation, such as: myresource -in file1 file2 ... -out file1 file2 …

where the name of the resource or operation to be invoked is followed by input and output file names introduced by the parameters "-in" and "-out" respectively. As usual, any conversion of these data files between the BDWorld representation using the RemoteData object and the form in which the resource reads or writes them is the responsibility of the wrapper, possibly supported by Resource Base and Resource Tools methods.

An additional form of resource command line is used, not to invoke an operation but to obtain values of the resource's metadata fields. This allows the wrapper to generate the XML description of these fields when it registers each resource.

An additional task, common to other approaches, is to define custom viewers for the resources to deal with user interaction. These viewers will be called with the appropriate hooks by the workflow manager.

4. Design issues to be resolved

During the design and implementation process, a number of general issues arose, which will be discussed individually.

4.1 Data interchange and persistence

The BDWorld communications architecture provides the BGI subsystem for communications between resources. The transport of data over the BGI requires serialisation of the data into a string: this may represent an XML document or it may be in some more specialised format such as Nexus, or may be a string with no specific internal structure.

In many cases data volumes are large (for example spatial data sets for climate patterns), so bandwidth is optimised by not passing the data itself between resources, but rather a handle for the data. This handle is represented either as an URI, as a path and filename on the resource file system, or in some cases as a primary key of a database record. In all cases, however, the primary representation of data or the data handle is as a 'Remote Data' document; this makes it much easier to manage large numbers of heterogeneous resources and data types in the workflow system than would otherwise have been the case.

The BGI implementation provides a flexible, generic mechanism for transfer of data between resources. A side effect of this generalisation is that resource operation signatures are poorly typed (essentially they are all the same), making it possible to pass inappropriate data (or data sets in the wrong order) to a resource without generating a compile-time or run-time error at the BGI level. The Resource Matcher module used by the workflow system has a role in preventing these errors, at least to the extent that the metadata describes the data sufficiently. If the resource itself cannot perform and report such checks in a useful manner, wrapper writers may wish to implement their own type checking internally. Future versions of BDWorld will include a mediation layer that the Resource Matcher implements by using richer metadata for every operation and data set to ensure that data being passed to a given resource is of the correct type.

When a data set is intended to be used at multiple points in a workflow, caching mechanisms of two types are employed: file-based and database-based. The lifetime of cached data depends on the nature of the data itself, and on institutional arrangements between the data supplier and resource provider. In the case of data which is often updated, such as localities from an online herbarium, the data source is queried each time a new workflow commences that uses localities data. This ensures that data used for modelling purposes is current. It is intended to extend the scope of the ephemeral cache to act as a fall-back for situations where the original source of the data is temporarily unavailable.

In other cases data is not expected to change very often, such as country boundaries used in map presentation. Efficiencies can be obtained by longer caching of these data and deferring to cached data when it is present in preference to possibly resource-intensive collection of data at origin.

[3] www.mygrid.org.uk

4.2 Exploratory workflow construction

The issues discussed in this paper presuppose an environment that is primarily aimed at helping users build and use scientific workflows. In some projects a completely different approach has been taken, perhaps most notably in BioMOBY [6]. In BioMOBY the emphasis is upon manipulation of *objects* of relevance to bioinformatics — for example, users can find related objects, and run applications that use these objects. This is a much more exploratory approach than initially envisaged in BDW, but the BioMOBY authors note that "logging and automated workflow discovery" would be a desirable extension to BioMOBY implementations. We plan, eventually, to support metadata-driven exploration in a manner similar to BioMOBY, but also to make it easy for users to integrate the results of their explorations into new workflows.

4.3 Mechanisms for user interaction

During enactment of a workflow, various kinds of interaction with the user may be required. For example, a user may be required to select from a list of options or to supply a search string, or he or she may be provided with a bitmap that presents results in a graphical form. Such interactions are easy to support using the Triana system. But there are situations that are potentially more problematical. Some of these situations, and solutions that we are considering, are as follows:

- A resource may have originally been designed to be used solely in an interactive manner, via a GUI, on the machine where it has been installed. One way this resource can be used is for it to lie outside the core BDW architecture: it could either be installed as a local tool, or it could be used remotely via some system such as VNC. No approach is ideal in this case: the BDWorld system loses control over some aspects of the workflow enactment and users may make mistakes (e.g. storing results in the wrong file) that will lead to error; also, users need to be running the appropriate operating system for local tools to be installed, and the VNC route will not scale up very well. But such techniques may be required *in extremis*.
- A 'fine-grained' interaction may be required at some points in a workflow, e.g. visualisation of a complex model and user selection of some item. This can either be implemented as specialised units for Triana, or by communication with the resource via the BGI. In the former case, a mechanism for delivery of such units to individual Triana instances will need to be defined, and we are limited by the computational power of the user's machine. In the latter case, data volumes may be too high for a mechanism such as the BGI to support, and a less general, high-performance extension of the BGI may need to be delivered.

The important thing to note is that it is undesirable to prevent the use of resources that cannot be made to conform directly with the BGI: new tools of relevance to BDWorld, but incompatible with it, will inevitably continue to be produced by research projects and software houses.

Acknowledgements

We thank BBSRC for funding the Biodiversity World project and the Triana team for making Triana available to us and providing us with technical support.

References

[1] Jones AC, White RJ, Pittas N, Gray WA, Sutton T, Xu X, Bromley O, Caithness N, Bisby FA, Fiddian NJ, Scoble M, Culham A and Williams P (2003). BiodiversityWorld: An Architecture for an Extensible Virtual Laboratory for Analysing Biodiversity Patterns. Proc. UK e-Science All Hands Meeting 2003, 759-765

[2] White RJ, Bisby FA, Caithness N, Sutton T, Brewer P, Williams P, Culham A, Scoble M, Jones AC, Gray WA, Fiddian NJ, Pittas N, Xu X, Bromley O and Valdes P (2003). The BiodiversityWorld Environment as an Extensible Virtual Laboratory for Analysing Biodiversity Patterns. Proc. UK e-Science All Hands Meeting 2003, 341-344

[3] Taylor I, Shields M, Wang I, Philp R (2003). Grid Enabling Applications Using Triana. Workshop on Grid Applications and Programming Tools, June 25, 2003, Seattle, USA. [www.cs.cf.ac.uk/user/I.J.Taylor/CV/Papers/GAPT_2003.pdf]

[4] Luger GF and Stubblefield WA (1993). Artificial Intelligence: Structures and Strategies for Complex Problem Solving, p335ff, Benjamin-Cummings.

[5] Stevens RD, Robinson AJ and Goble CA (2003). myGrid: personalised bioinformatics on the information grid. Bioinformatics 19, Suppl. 1, i302-i304, Oxford U.P.

[6] Wilkinson MD and Links M (2002). BioMOBY: An Open Source Biological Web Services Proposal. Briefings in Bioinformatics 4, 1-11.