

Engineering considerations for biodiversity software

Robert A. Morris ⁽⁹⁾, Mathew Passell ⁽⁹⁾, Jun Wan ⁽⁹⁾, Robert D. Stevenson ⁽¹⁰⁾ and William Haber ⁽¹¹⁾

Abstract

We describe how object oriented design and programming, together with object databases, support applications that require diversity in their data structures reflecting diversity in the description of the data. A three tier web-based architecture permits flexible multiple views on the data. With working instances of an electronic field guide and of applications that federate distributed biodiversity data, we show how XML and object technologies can ease the burden of biologists who must prepare descriptive and diagnostic data for inclusion in web-accessible database.

Introduction

Biodiversity software encompasses a wide range of applications including the maintenance of specimen records, analysis of phylogenies, examination of biogeographic relationships, and recording of ecological observations (see Biosis, 2000, Geocities, 2000, and Lampinen, 2000 for lists of free and commercial packages). The rapid development in power and sophistication of biodiversity software in the last 15 years mirrors progress in the broader software industry. All software is moving from single user platforms to Web based tools, from text to graphical user interfaces and from custom made software built from the ground up to applications layered on top of commercial or open source components.

Biodiversity software concentrates at the species level because species are the leaves of the hierarchical grouping system called the 'taxonomic tree' used by scientists to classify life forms. Despite limitations to the species concept (Futuyma, 1998), the classification system is well established in the scientific community, having been used since Carl Linnaeus invented it over 250 years ago. Therefore is it not surprising that a significant portion of biodiversity software deals with the management of taxonomic information. Examples include programs that help manage collections of specimens such as Biota (Colwell, 1996) and Biolink (Shattuck and Fitzsimmons, 2000) from which taxonomists describe and name species, construct keys to differentiate species (Dallwitz, 2000) or document the tree itself (Maddison, 2000). Common to all of these programs and to many other efforts to share biodiversity information is the species page (also called the homepage, species summary or species treatment) in which biologists present basic information about a species.

Below we describe a biodiversity software application we are developing called the Electronic Field Guide (EFG). The EFG has elements common to biodiversity software listed above but different goals (see below). There are many efforts within scientific and environmental communities to establish standards and way of linking biodiversity information across multiple databases. Among these efforts are those of the Taxonomic Database Working Group <http://www.tdwg.org/>, the U.S. National Science Foundation's Long Term Ecological Research Sites <http://www.lternet.edu/informatics/>, those of governmental agencies such as the National Biological Information Infrastructure, NBII <http://www.nbio.gov/> and <http://www.nbio.gov/home/partner/bioeco/index.html>; the Global Biodiversity Informatics Facility (GBIF) <http://www.gbif.org/>); Cornell's Laboratory of Ornithology Citizen Science projects <http://birds.cornell.edu/citsci/>, the Nature Mapping Program <http://>

(9) University of Massachusetts, Department of Computer Science, 100 Morrissey Boulevard, Boston MA 02125, USA, ram@cs.umb.edu

(10) University of Massachusetts, Department of Biology, 100 Morrissey Boulevard, Boston MA 02125, USA

(11) Missouri Botanical Garden, P.O. Box 299, St. Louis, Missouri 63166, USA

www.fish.washington.edu/naturemapping/; and Discover Life in America <http://www.discoverlife.org>. In the second part of the paper we discuss how XML (the eXtensible Markup Language) now being adopted in the Web community as a replacement to HTML is likely to help with the federation of information across multiple heterogeneous data sources.

The UMASS-Boston Electronic Field Guide Project, UMB-EFG (Stevenson and Morris, 2000) provides a web-accessible distributed object-oriented database for the identification of biological specimens from field observations. The data, including both taxonomic and environmental or ecological data, will aid in identification by building a context for each observation. As observation data accumulates, larger-scale ecological studies can be carried out using the data. UMB-EFG is being constructed and populated under The EFG project has recently been expanded to encompass investigation of a number of issues and solutions under discussion in the eco-informatics community, including the use of XML for federation of data from disparate distributed database, as well as for more common tasks such as data exchange and system configuration. This paper describes our engineering approach to the building of these systems and reports on their current status.

Field guides and descriptive data: object-oriented representations

Central to the majority of field studies in biology is the correct scientific identification of species. People learn from others or use field guides if possible, but for most groups identification is accomplished with keys constructed by specialists with knowledge and experience in a taxonomic group. The process begins with the collection of specimens in the field. Taxonomists and systematists then prepare, study and catalogue the specimens, usually in academic institutions or natural history museums. Finally, written descriptions are published and a name given to each new species discovered. Some taxonomists work only in the laboratory, obtaining the specimens from field collectors.

Most paper field guides are devoted either to a specific collection of taxa, e.g. birds, trees, wild-flowers, etc., of a specific geographic region. In most cases, a field guide user who is interested in ecological interactions will require several different field guides. For example, a guide to butterflies may have some narrative identifying the host and nectar plants of a particular butterfly species, but it would give little help in identifying that plant (which might in turn help the reader identify the butterfly). An electronic field guide on contemporary computers (or the web) can easily hold data on a wide variety of taxa, but a data representation issue immediately arises: descriptive characters appropriate to one group of taxa may have little to do with a radically different group. For example, plants have no wing spots since they have no wings, and butterflies have no leaves to be characterized as simple or compound. Therefore, to represent both organisms in a traditional relational database one must either accept large sparse tables or manage very complex joins on the ecological relationships. Essentially, the diversity of life is not amenable to a single description.

Object-oriented Database Management Systems (OODBMSs) (King, 1997) are a solution to the problem of representation of biodiversity, because their data is self-describing. (This is also true of XML, of which more later). Saarenmaa et al. (1995) have observed in detail how object-oriented techniques in general, and OODBMS in particular are well suited to taxonomic databases. However, they reported that with technologies then available, they were unable to usefully create taxa as classes, rather than instances of a single class. In a database with a huge number of taxa, class loading overhead would still make this impractical today. In any case this approach might model specimen collections well, but it does not seem appropriate to a field guide, where it would result in one instance per class. Instead, we model an author's treatments of a large group, e.g. a family, as a class, and model individual taxa as instances. The cost of this is that a species described in an electronic field guide in several different treatments, e.g. by different authors for different locations, are not in the same object-oriented class. We discuss our approach to this issue below. We have implemented the UMASS-Boston Electronic Field Guide (UMB-EFG, or just EFG) on eXcelon Corporation's Object Store product (eXcelon, 2000). This OODBMS is a persistent store for Java (or C++) classes, and we describe next how we create such classes, along with our design requirements for author-friendliness, i.e. criteria by which the authors of descriptive data and keys are kept

isolated from the technology. We note in passing that object/relational database management systems (ORDBMSs) (Grimes, 1998) may well provide the support we require, but did not have mature programming interfaces at the time we began coding.

A design requirement of the EFG is that the system should be fundamentally ignorant of the nature of descriptive data. If an author chose to offer characters of restaurants of Boston, our software would produce a meaningful and useful restaurant guide instead of a field guide to the butterflies of Costa Rica, our initial target.

Diverse data such as we described above is called *semi-structured data* in recent literature (see Abiteboul et al., 2000). This means that it need not have a precise database schema such as would be found in a relational database. Pure OODB's are a special case, but of course so is a relational database (Abiteboul et al. *op. cit.*). This property makes it particularly easy for us to import diverse descriptive data and combine it in a single database. The taxon abstraction we use is a software construction called a *JavaBean*. (Sun, 2000a). Beans differ from other Java classes in having particularly convenient *introspection*. The introspection interface allows programs to inquire of the bean — in our case a collection of taxa having a common set of descriptive characters — what its properties and behavior are. This frees us from requiring advance knowledge of that behavior. As a simple example, when the user selects a group of taxa, e.g. butterflies of Costa Rica, we create, on the fly, a biologist-friendly search facility comprising a typical web form with pull-down menus for each character, the names of the character having been gained by introspection and the possible character values by inquiry into the database. (Here we mention 'biologist-friendly' because we will also describe below how our architecture supports flexible user interfaces by separating the UI from the rest of the system. In particular, we'll describe experiments in visual keys more suitable for amateur users.) A JavaBean must be compiled and loaded into the current Java Virtual Machine, so in essence we are generating and compiling source code on-the-fly when we import biological taxon treatments and we do this with some unsupported Java compiler classes available from Sun. This being a somewhat fragile approach, we are presently designing a different mechanism based on Java Map (Sun, 2000b) interfaces, which also can describe and manipulate objects based on their properties, and so are good candidates for a software model of a taxon.

Architecture overview

We have a typical three-tier Web application with Java servlets as middleware forwarding questions to an eXcelon Corp. ObjectStore OODB. HTML forms and Java Applets pass queries to the servlets, which transform them into queries suitable for the ObjectStore backend. After retrieving the data, the servlets build html pages, or send data to the applets, for presentation to the user. Although we operate our ObjectStore and servlets on the same host, this is completely unnecessary. In addition, ObjectStore is itself a distributed client-server system and various pieces of the database could be scattered around the Internet with no change to our architecture except to add resource discovery mechanism to find the distributed data. In a symmetric fashion, because servlets accept connections on internet IP ports, other clients than our own front end can forward queries to our servlets. See Nakhimovsky and Myers (1999) for an introduction to three-tier applications.

Importing data

We do not require an author to understand object-oriented technologies. Most biologists keep their descriptive data in software that is, or rests on, standard table-based databases or in a spreadsheet. Typical systems are Excel, Microsoft Access, FileMaker, and specialized systems such as Biota (Colwell, 1996), which uses 4D ODBC support (4D, Inc., 2000). Most of these products can respond to SQL queries along an ODBC connection (Microsoft, 1999) and in turn can be accessed in Java by the JDBC-ODBC bridge (Sun, 1997). More sophisticated systems based on Oracle can use JDBC natively. Our code uses this bridge to import database field names (i.e. character names). We must also attach the data at the appropriate place in the taxonomic tree, because in most cases the author will deliver data about a group of taxa,

e.g. a family in a particular locale, without a complete taxonomy. To accomplish the attachment we require a simple XML description of the common taxonomic hierarchy above all the taxa in the data source.

This import strategy is convenient, but leads to several problems. Most notably, the order of the delivery of fields is not determined by the ODBC protocol. Typically the order is that in which the fields were created, but that is often not the order in which the biologist may wish them displayed. Our solution, not fully implemented, is to specify the character order in metadata that is also represented in XML. A second difficulty is that it is possible to specify field names in many databases that do not yield legal Java identifiers (e.g. they contain white space). We convert illegal characters to underscores, but this is not a robust solution, since it may not allow going backwards from the OODB to the original. XML metadata specifying a one-to-one mapping of illegal characters to not-necessarily easily readable Unicode characters in Java identifiers would suffice, but would probably require tools to make the Java identifiers readable for software maintenance. Table 1 shows a typical XML metadata file.

Table 1. Sample Metadata file

<EFGMetadata MetaDataID='Test'>
<EFGField>
<name>AntennaColor</name>
<type>morphological</type>
<weight>0</weight>
<dataCount>single</dataCount>
<dataType>simple data</dataType>
<javaType>java.lang.String</javaType>
</EFGField>
<EFGField>
<name>Habitat</name>
<type>ecological</type>
<weight>0</weight>
<dataCount>multiple</dataCount>
<dataType>simple data</dataType>
<javaType>java.lang.String</javaType>
</EFGField>
<name>Similar_Species</name>
<type>ecological</type>
<weight>0</weight>
<dataCount>multiple</dataCount>
<dataType>taxonomic reference</dataType>
<javaType>java.lang.String</javaType>
</EFGField>
...
</EFGMetadata>

Sample Data File associated with the above Metadata file

Table 2.

<EFGImportDocument MetaDataID='Test'>
<CommonPath>
<Domain>Eukaryotes</Domain>
<Kingdom>Animalia</Kingdom>
<Phylum>Arthropoda</Phylum>
<Class>Insecta</Class>
<Order>Lepidoptera</Order>
<Family>Nymphalidae</Family>
<Subfamily>Ithomiinae</Subfamily>
</CommonPath>
<TaxonList>
<Taxon>
<EFGPath>
<Genus>Ithomia</Genus>
<Species>heraldica</Species>
</EFGPath>
<CharacterData>
<ForeWingSpot>yes</ForeWingSpot>
<AntennaColor>orange</AntennaColor>
...
</CharacterData>
</Taxon>
<Taxon>
<EFGPath>
<Genus>Mechanitis</Genus>
<Species>polymnia</Species>
</EFGPath>
<CharacterData>
...
</CharacterData>
</Taxon>
...
<Taxon>
...
</Taxon>
</TaxonList>
</EFGImportDocument>

To present to a user with data arising from several heterogeneous sources there are fundamentally two approaches. One may build an integrated database and import each source, or one may build a federated view in response to a particular query that is executed—possibly after rewriting—against all of the data sources. The UMB-EFG architecture supports both, but our current implementation is only of the former. Later we describe some preliminary work in federation in which the UMB-EFG could potentially be one of the data sources, rather than a federating data consumer. However, even in our current implementation, any application anywhere on the Internet could make an IP connection to our Java servlet middleware and make queries in the same way our own HTML forms and Java servlet front-ends now do.

When we import data from an author's database we produce intermediate XML with the character data and such taxonomy as is described in the data source, and we require the author to provide some simple external taxonomy in XML form, namely `<CommonPath>` element in the example in Table 2. This latter specifies a position in the taxonomic tree at which the group is to be attached, specified as an edge-labeled path in the taxonomic tree, with edges labeled by the taxon level and nodes by the taxon name. The edge labels, i.e. the taxonomic level, are generated automatically—the author need only supply the taxon name. Authors can use simple available tools, such as Microsoft XML Notepad, to provide this CommonPath information. Edge-labeled digraphs have been considered previously for taxonomic data representation (Zhong et al., 1996), (NCGR, 2000). All data in the source must have taxonomy specified relative to the terminal taxon of that path. For example, if the source provides character values for the Ithomid butterflies, each record would provide genus and species. Typical XML generated by the importer is illustrated briefly in Table 2. From XML representation to JavaBeans is a transformation well-supported by currently available Java classes.

In Ithomid butterfly example, the imported data would normally be expected to have a field labeled Genus and one labeled Species and the data will be created in the taxonomic tree by subtrees under the Ithomiinae node, each subtree having edges labeled *Genus* and *Species* and with nodes appropriately named from the values in the imported data. Also, where appropriate, we treat various life forms (e.g. larva, pupa, etc.), and also each sex if the data identifies them as different, as though they were at a taxonomic level below species.

In addition, we have metadata for each character that comprises:

- The character name (e.g. *similarSpecies*, *foreWingSpotPresent*)
- A character type (presently either *morphological* or *ecological*)
- A numeric weight by which the author has ranked the character as to its importance in identifying an organism (See Dallwitz (1999b) for a discussion of this issue.)
- A string describing whether there may be multiple data in the same field
- A data type, presently one of *simple data*, *image filename*, *sound filename*, or *taxon reference*. A taxon reference is a reference to another taxon already in, or to be inserted in the database
- A Java data type.

Ultimately, we intend to treat this metadata with an XML Schema (W3C, 2000a) but currently the Java support for XML Schema is not mature. However, at this writing many other XML tools are.

User interfaces

Our three-tier interface allows us to support a variety of user interfaces. We presently have three in place, requiring increasing levels of biological sophistication.

The first of these is a purely visual interface to the Ithomid butterflies of Costa Rica, designed by one of us (Haber) with extensive field experience in the subject. In this interface the butterflies are first divided into four visually distinctive groups ('Tiger', 'Clearwing', 'Yellow and Black', and 'Translucent Gold'). We display a conventional tree browser, functionally identical to the file system browser found on Windows but instead of folder and file icons, the

icons are thumbnail images of actual Ithomids representative of the group. As with a file browser, the user can click to expand one of the icons to another series of 3-4 butterflies representing a more subtle visual distinction within the group. This continues recursively about four deep and usually 3-4 wide. At leaf nodes of this tree, clicking brings a descriptive page for the species identified. There may be several paths to the same species, so internally this UI is represented not by a tree but by a directed graph (digraph). Digraphs are familiar data structures that have found related application in biological keys (Beaman et al., 1999), (Zhong et al., 1996), (NCGR, 2000). Note that key digraphs are not, in general, trees. That is, there may be several paths to the same taxon. We believe that a purely visual interface corresponds closely to the way amateurs use field guides, because we find that amateurs quickly get lost in traditional dichotomous keys and their generalizations. The biggest issue with this or any key is to verify that the identification is in fact correct. As an aid to this, our species description pages also contain links to similar species so that the user may compare their chosen species to ones that the author warns may be confused with it.

The second interface is a tree browser that follows the standard taxonomic tree, similarly to the Tree of Life project (Maddison, 2000). That system moves from web page to web page in the tree traversal, whereas we use the same Java tree browsing classes underlying the visual key. Hence, as for a file system browser, we keep everything on the same screen. The population of our database is presently too small for us to decide whether the attendant required scrolling would be counterproductive in a collection as large as the Tree of Life.

Finally, we have a pure html form-based character-value interface. The forms are constructed on-the-fly from the database, and the middleware builds and serves the form to the browser. Each form field offers the user the possible character states. The user can select the maximum number of species they wish to have offered that meet the current character value choices, and if the system finds more, the user is invited to limit the output by further choice from another character. In our present implementation we do not illustrate those choices as do a number of systems, but this is not precluded by our architecture. Without such illustrations, our interface is suitable mainly for users with some training. The use of characters with a finite number of states complicates naturally continuous numeric characters with ranges (Dallwitz, 1999a).

Because our species description pages are made up by the middleware in all cases, it is not difficult for us to produce XML instead of HTML for description pages as we do in the present implementation of our servlet middleware. Doing so has well-known advantages, some of which we have been exploring even though we do not yet output XML. These are discussed in subsequent sections.

Federating and transforming XML data

Here we describe our preliminary application of XML technologies to some of the issues arising in biodiversity software. Our experience as yet is too small, and the tools too new, for us to offer many engineering insights, so this section is devoted to discussing the capabilities of our prototypes and our current directions. All the applications in this section may be reached from the link 'recent XML work' on our home page <http://www.cs.umb.edu/efg>. This work is supported by a subcontract from the University of Kansas Biodiversity Research Center under their NSF grant KDI-9873021.

The range and quantity of biodiversity data on the web is large and rapidly growing (UNO, 2000). It is presently difficult to combine data from this large collection of sources for several reasons. First, few such sources publish or support any API to the underlying databases. Therefore, anyone who wishes to interrogate such data programmatically must reverse engineer the arguments passed by the publisher's web interface.

A second impediment to data federation, despite the explosion of XML support, is that the pages that are returned are generally HTML and so contain detailed presentation markup and little, if any, markup of structural or biologically descriptive utility. The Tree of Life project (Maddison, 2000) embeds special markup in support of its own web crawling engine,

and this can serve some of the requirement for metadata that can be addressed with XML. We are aware of a number of projects imminent or underway to serve XML, including the Integrated Taxonomic Information System (ITIS, 2000), (ITIS^{ca}, 2000), a joint project of the US Department of Agriculture and the Canadian Ministry of Agriculture. ITIS^{ca} already has experimental XML service on its site. The Biodiversity Research Center of the University of Kansas has implemented a gateway from Z39.50 servers to XML (Vieglais, 2000), about which we say more below. Indeed, anecdotally we understand that many biodiversity database maintainers and designers intend to serve XML.

XML is widely accepted as a data exchange language and for the specification of metadata and of configuration. We described above some of this use for the UMB-EFG itself. Early public uses of XML as a replacement for HTML for presentation focused on transforming XML to HTML in the browser (supported to date mainly in Microsoft Internet Explorer 5 (MSIE5), and we describe some demonstrations of that below. More importantly, XML can also be used for federating data from heterogeneous data sources (Abiteboul et al., 2000), (Baru et al., 2000). We will next sketch our experiments with these applications of XML to biodiversity software.

XML holds the promise of extremely flexible user interface configuration. Either at the server or in an XML-aware browser, a separate stylesheet is written in XSLT, the transformation language of the XSL stylesheet language (W3C, 2000a). This transforms the XML representation of, say, a descriptive taxon page, into HTML in a manner suited to the browser user or to the policies of the host that is forwarding the XML page (which may well be different from the host holding the original data). For example, among the XML demonstrations linked on the project page (Stevenson and Morris, 2000), we have one in which a number of static species description pages from various sources may be displayed with a number of sometimes widely varying styles (including one that shows the raw XML). When using MSIE5 the user need only push a button in the control panel to instantly change style sheets, the redisplay being handled in the browser itself and not requiring any return to the server for the new view.

Because XSLT is very general, it is possible to make mini-applications simply by transforming to a restricted view. For example, one of our applications accepts a list of (suitably marked) XML species description pages and produces a table of references in each of those pages to other taxa. This application looks like a database operation, but in fact is carried out entirely in whatever is executing the XSLT (in our case, the MSIE5 browser).

Another XSLT application we show is a bilingual display. With a single button click in a control frame, a species page is toggled between a Spanish and an English view. Again all transformations are done by the browser. In this application, text in both languages is kept in a single file (easing maintenance), with a language attribute on each text element. Under control of a simple Java Script program in the control frame, a single variable (*'currentLanguage'*) is toggled and a single XSLT function displays an element only if its language attribute matches *currentLanguage*.

Among applications of XML, the promise of support of database federation is the most distant from the conventional applications of XML, but it is also the most challenging and interesting application. There is substantial advantage to standardized DTD's (or better, XML Schemas when they are stable) to combine data from various sources (Baru et al., 1999), (Baru et al., 2000). But recent research suggests that suitable schemas can be inferred from the sources themselves (Ludäscher et al., 1999), (Abiteboul et al., 2000). For the moment we use a fixed DTD and code static XML species description pages to that somewhat spare DTD. Those pages lack complete taxonomy, containing, as is typical, only the genus and species name. We have built a distributed federator based on (a very small part of) that known DTD, and knowledge of the query syntax of *zportal*, an XML/Z39.50 gateway built in The Species Analysis (TSA) project (Vieglais, 2000). Z39.50 is an international data exchange standard in wide use in government agencies and more recently adopted by many museums for their collection data. Because we use the *zportal*, details of Z39.50 are unimportant here, but they are discussed at the technical pages (on a link named 'Z.X') of the TSA site. In our federator, a JavaScript control panel on the application page looks into the static XML species page to

find the scientific name of the species described. This is sent to a mediator on our web server, a Java servlet that understands both the zportal query syntax and the location of taxonomic resources on the web. In this case, these resources comprise a small fixed collection of Z39.50 servers that offer taxonomic authority for specific collections of taxa. The mediating servlet queries each source until it finds the taxonomy for the given species, translates the XML returned by zportal into something user-friendly, and returns a combined page to the browser. (The zportal software returns XML, but based on a DTD whose element tags are simply XML rendering of the underlying numerically coded Z39.50 record tags).

The federation application described above finesses a number of difficult points that are the subject of our (and others') future work. The problem of resource discovery, e.g. finding taxonomic authority servers, is a deep and interesting one that appears in many similar contexts. So also does query rewriting in case the sources require different query syntax, and record rewriting in case the sources have different record semantics

Finally we mention an interesting issue that is the subject of work we are just starting: circumscription control—the control of the forwarding of sensitive data obtained from trusting sources. Some biodiversity data is sensitive (e.g. the precise location of specimens of endangered species). Consequently this data is often made available only to users who are trusted by the data holder not to abuse it. Unfortunately, most existing biodiversity data is held in databases with granularity of circumscription no finer than the entire database itself. No provision is in place to prevent access to sensitive data on a field-by-field basis or even a record-by-record basis. For example, specimen location data is sensitive only for endangered species, so circumscribing all location data is debilitating to many legitimate users and applications. To address this issue we are designing a *circumscription broker*. This is a set of protocols and software by which a data source can specify its circumscription policies and trust the broker to enforce them, thereby leaving the source able to serve entire records at will. Because the broker is acting on behalf of a database mediator, which is disassembling and reassembling records for the federated view in any case, the task of enforcing the circumscription belongs at the broker/mediator and mediator/query-response interfaces.

Conclusion

Biodiversity software is best implemented with software tools designed for dealing with data diversity. The engineering details of these tools can be hidden from the biologist who must use or populate the systems built around them. Java and XML technologies prove both suitable for this purpose, and highly synergistic with each other. Using them we have demonstrated an extensible web-based Electronic Field Guide and how to build applications that federate data from distributed sources around the internet.

Acknowledgements

Robert Morris, Matthew Passell, Robert Stevenson, and William Haber are partially supported under NSF Grant *DBI-9808462* from the US National Science Foundation.

Wan is partially supported under grant KDI-9873021 to the Biodiversity Research Center at the University of Kansas.

This paper is an extension of a presentation given by Morris and Stevenson at the International Congress of Entomology, Iguassu, Brazil, August 2000.

References

- 4D ODBC Driver, 2000. <http://www.4d.com/products/odbcdriver.html>.
- Abiteboul, S., Buneman, P., Suci, 2000. Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann Series in Data Management Systems, Morgan Kaufmann, San Francisco.
- Baru, C., Gupta, A., Ludäscher, B., Marciano, R., Papakonstantinou, Y., Velikhov, P., 1999. XML-Based Information Mediation with MIX. Exhibitions Program of ACM SIGMOD 99. Also available at <http://www.db.ucsd.edu/publications/vamp.pdf>
- Baru, C., Ludäscher, B., Papakonstantinou, Y., Velikhov, P., Vianu, V., 2000. Features and Requirements for an XML View Definition Language: Lessons from XML Information Mediation. Principles Of Database Systems (PODS) 2000. Also available at <http://www.db.ucsd.edu/publications/xmas.html>.
- Beaman, J., Luo, Y., Pramanik, S., Zhong, Y., February, 1999. HICLAS: A taxonomic database systems for comparing biological classification and phylogenetic trees. *Bioinformatics Journal*.
- Biosis and Zoological Record's Internet Resource Guide for Zoology Software, 2000. <http://www.york.biosis.org/zrdocs/zoolinfo/software.htm>
- Colwell, R., 1996. Biota: The Biodiversity Database Manager. Sinauer Press, Sunderland Massachusetts. See also <http://viceroy.eeb.uconn.edu/Biota>
- Dallwitz, M.J., 1999. Desirable Attributes for Interactive Identification Programs. <http://biodiversity.uno.edu/delta/www/idcrit.htm>
- Dallwitz, M.J., 1999. Introduction to Data requirements for Natural-language Descriptions and Identification. <http://biodiversity.uno.edu/delta/www/descdata.htm>
- Dallwitz, M.J., 2000. A Comparison of Interactive Identification Programs. <http://www.biodiversity.uno.edu/delta/www/comparison.htm>
- Object Store Product Description, 2000. <http://www.exceloncorp.com/products/objectstore.html>
- Futuyma, D.J., 1998. *Evolutionary Biology*, 3rd Edition. Sinauer Associates, Massachusetts.
- Digital Taxonomy Software, 2000. <http://www.geocities.com/RainForest/Vines/8695/>
- Grimes, S., April, 1998. Modeling Object/Relational Databases. DBMS, <http://www.dbmsmag.com/9804d13.html>
- Integrated Taxonomic Information System, 2000. <http://www.itis.usda.gov/plantproj/itis>
- ITIS*ca - Canadian version of ITIS Integrated Taxonomic Information System, 2000. <http://res.agr.ca/itis/>
- King, N.H., June, 1997. Object DBMSs: Now or Never. DBMS, <http://www.dbmsmag.com/9707d13.html>
- Lampinen, R., October, 2000. Cartographic Links For Botanists. <http://www.helsinki.fi/~rlampine/cartogr.html>
- Ludäscher, B., Papakonstantinou, Y., Velikhov, P., Vianu, V., 1999. View Definition and DTD Inference for XML. Post-ICDT Workshop on Query Processing for Semistructured Data and

- Non-Standard Data Formats. Also available at <http://www.db.ucsd.edu/publications/icdt-ws-99.pdf>
- Maddison, D. (Coordinator and Ed.), 2000. The Tree of Life, <http://phylogeny.arizona.edu/tree/phylogeny.html>
- Microsoft Corporation, 1999. Microsoft ODBC. <http://www.microsoft.com/data/odbc/>
- Nakhimovsky, A., Myers, T., 1999. Professional Java XML Programming with Servlets and JSP. Wrox Press, Birmingham, UK.
- National Center for Genome Resources NCGR Taxonomy Project, 2000. <http://www.ncgr.org/research/sequence/taxonomy.html>
- Papakonstantinou, Y., Velikhov, P., 1999. Enhancing Semistructured Data Mediators with Document Type Definitions. Data Engineering 99. Also available at <http://www.db.ucsd.edu/publications/icde99.pdf>
- Saarenmaa, H., Leppäjärvi, S., Perttunen, J., Saarikko, J., 1995. Object-oriented taxonomic biodiversity databases on the World Wide Web. In: Kempf, A., Saarenmaa, H. (Eds). Internet Applications and Electronic Information Resources in Forestry and Environmental Sciences. Workshop at the European Forest Institute, Joensuu, Finland, August 1-5, 1995. EFI Proceedings 3. Also available at <http://www.efi.fi/~saarenma/oobdwww-nature-latest.htm>
- Shattuck, S., Fitzsimmons, N.J., 2000. BioLink®The Biodiversity Information Management System. CSIRO Publishing, Collingwood Victoria 3066, Australia. See also <http://www.ento.csiro.au/biolink/index.html>
- Stevenson, R.D., Morris, R.A. (PIs). The UMASS-Boston Electronic Field Guide Project. <http://www.cs.umb.edu/efg>
- Sun Microsystems, 1997. JDBC-ODBC Bridge Enhancements. <http://java.sun.com/products/jdk/1.2/docs/guide/jdbc/bridge.html>
- Sun Microsystems, 2000a. JavaBeans Tutorial , Part 1. <http://developer.java.sun.com/developer/onlineTraining/Beans/Beans1/index.html>
- Sun Microsystems, 2000b Java Map Interface API, 2000. <http://java.sun.com/products/jdk/1.2/docs/api/java/util/Map.html>
- The Biodiversity and Biological Collections Web Server, 2000. <http://biodiversity.uno.edu>
- Vieglais, D., 2000. The Species Analyst. <http://habanero.nhm.ukans.edu/>. For description of the portal interface, go to the Z.X section of the site.
- World Wide Web Consortium, 2000. XML Schema. <http://www.w3.org/XML/Schema>
- World Wide Web Consortium, 2000. Extensible Style Sheet Language. <http://www.w3.org/Style/XSL/>
- Zhong, Y., Jung, S., Pramanik, S., Beaman, J.H., May, 1996. Data model and comparison and query methods for interacting classifications in a taxonomic database. Taxon, 45(2). Also available at <ftp://ftp.cps.msu.edu/pub/hiclas/paper1/paper1.txt>

Towards a global biological information infrastructure

**Challenges, opportunities,
synergies, and the role of entomology**

**Edited by:
H. Saarenmaa and E. S. Nielsen †**

**Project manager:
Hannu Saarenmaa
European Environment Agency**



Legal notice

Neither the European Environment Agency nor any person or company acting on behalf of the Agency is responsible for the use that may be made of the information contained in this report.

A great deal of additional information on the European Union is available on the Internet. It can be accessed through the Europa server (<http://europa.eu.int>).

© EEA, Copenhagen, 2002 and
© Authors of their respective articles.

Reproduction is authorised provided the source is acknowledged.



European Environment Agency
Kongens Nytorv 6
DK-1050 Copenhagen K
Tel. (45) 33 36 71 00
Fax (45) 33 36 71 99
E-mail: eea@eea.eu.int
Internet: <http://www.eea.eu.int>

Contents

Preface: Towards a global biological information infrastructure: Challenges, opportunities, synergies, and the role of entomology H. Saarenmaa	4
The Tree of Life project: A multi-authored, distributed Internet project containing information about phylogeny and biodiversity D.R. Maddison, W.P. Maddison, J. Frumkin and K.-S. Schulz	5
Issues of quality control in large, mixed-origin entomological databases J. Soberón , L. Arriaga and L. Lara	15
Interactive identification using the Internet M. Dallwitz, T.A. Paine and E.J. Zurcher	23
New approaches to creating global species databases in entomology M.J. Scoble	34
An information infrastructure for German insect collections including multimedia and GIS tools K.-H. Lampe and K. Riede	43
Engineering considerations for biodiversity software R.A. Morris, M. Passell, J. Wan, R.D. Stevenson and W. Haber.	49
Technological opportunitites and challenges in building a global biological information infrastructure H. Saarenmaa	60