A Distributed Annotation System

Authors: Robin Dowell

One goal of any genome project is the elucidation of the primary sequence of DNA contained within a given species. While the availability of the primary sequence itself is valuable, it does not reach its full potential until it has been annotated. Generally defined, annotation is descriptive information or commentary added to text, in this case genomic sequence. Without a mechanism for collecting, recording, and disseminating community-based annotation, a valuable source of information is severely diminished. In this report I outline the design and implementation of a Distributed Annotation System (DAS). DAS allows sequence annotations to be decentralized among multiple third-party annotators and integrated on an as-needed basis by client-side software. A single server, designated the "reference server," provides essential structural information about the genome: the physical map which relates one sequence to another, the DNA sequence for each entry, and the authorship information. Multiple sites then act as third-party "annotation servers." Using a web browser-like client application, researchers can interrogate one or more annotation servers to retrieve features in a region of interest.... Read complete abstract on page 2.
A Distributed Annotation System

Complete Abstract:

One goal of any genome project is the elucidation of the primary sequence of DNA contained within a given species. While the availability of the primary sequence itself is valuable, it does not reach its full potential until it has been annotated. Generally defined, annotation is descriptive information or commentary added to text, in this case genomic sequence. Without a mechanism for collecting, recording, and disseminating community-based annotation, a valuable source of information is severely diminished. In this report I outline the design and implementation of a Distributed Annotation System (DAS). DAS allows sequence annotations to be decentralized among multiple third-party annotators and integrated on an as-needed basis by client-side software. A single server, designated the "reference server," provides essential structural information about the genome: the physical map which relates one sequence to another, the DNA sequence for each entry, and the authorship information. Multiple sites then act as third-party "annotation servers." Using a web browser-like client application, researchers can interrogate one or more annotation servers to retrieve features in a region of interest. The servers return the results using a standard data format, allowing the annotation browser to integrate the information and display a graphical representation of the data. When an annotation is particularly interesting, they can easily return to the originating database for additional information. In short, DAS provides an indexing mechanism between sequence databases.

This technical report is available at Washington University Open Scholarship: https://openscholarship.wustl.edu/cse_research/250
A Distributed Annotation System

Robin Dowell

WUCS-01-07

April 2001

Department of Computer Science
Washington University
Campus Box 1045
One Brookings Drive
St. Louis MO 63130
Masters Project Report

A Distributed Annotation System

Robin Dowell

Computer Science
(e-mail: robin@genetics.wustl.edu)

Committee:
Michael Brent
Sean Eddy
Gary Stormo

21 February 2001
Abstract

One goal of any genome project is the elucidation of the primary sequence of DNA contained within a given species. While the availability of the primary sequence itself is valuable, it does not reach its full potential until it has been annotated. Generally defined, annotation is descriptive information or commentary added to text, in this case genomic sequence. Without a mechanism for collecting, recording, and disseminating community-based annotation, a valuable source of information is severely diminished.

In this report I outline the design and implementation of a Distributed Annotation System (DAS). DAS allows sequence annotations to be decentralized among multiple third-party annotators and integrated on an as-needed basis by client-side software. A single server, designated the "reference server," provides essential structural information about the genome: the physical map which relates one sequence to another, the DNA sequence for each entry, and the authorship information. Multiple sites then act as third-party "annotation servers." Using a web browser-like client application, researchers can interrogate one or more annotation servers to retrieve features in a region of interest. The servers return the results using a standard data format, allowing the annotation browser to integrate the information and display a graphical representation of the data. When an annotation is particularly interesting, they can easily return to the originating database for additional information. In short, DAS provides an indexing mechanism between sequence databases.
Contents

1 Introduction ........................................ 5
  1.1 Annotation ...................................... 5
  1.2 Sequence Annotation ........................... 6
    1.2.1 Repositories .............................. 7
    1.2.2 Curatorial Databases .................... 7
    1.2.3 Networking .............................. 9
    1.2.4 Visualization .......................... 10
  1.3 The Distributed Annotation System .......... 11
  1.4 Project Collaborators ........................ 16

2 System Design .................................... 17
  2.1 Reference Sequence ........................... 17
  2.2 Annotation Servers ........................... 18
  2.3 DAS XML Specification ....................... 19

3 Implementations .................................. 23
  3.1 Prototype I .................................. 24
  3.2 Servers ...................................... 26
  3.3 Prototype II .................................. 28
  3.4 Prototype III ................................ 33
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 Geodesic</td>
<td>34</td>
</tr>
<tr>
<td>4.1 Hydra</td>
<td>34</td>
</tr>
<tr>
<td>4.2 Graphical User Interface</td>
<td>37</td>
</tr>
<tr>
<td>5 Comments and Future Work</td>
<td>42</td>
</tr>
<tr>
<td>5.1 Specification</td>
<td>43</td>
</tr>
<tr>
<td>5.2 Servers</td>
<td>45</td>
</tr>
<tr>
<td>5.3 Clients</td>
<td>46</td>
</tr>
<tr>
<td>A Specification Summary</td>
<td>49</td>
</tr>
<tr>
<td>A.1 The Queries</td>
<td>49</td>
</tr>
<tr>
<td>A.2 Feature Types and Categories</td>
<td>62</td>
</tr>
<tr>
<td>B ACeDB Server HOWTO</td>
<td>66</td>
</tr>
<tr>
<td>B.1 The Database</td>
<td>66</td>
</tr>
<tr>
<td>B.2 The Translator Script</td>
<td>67</td>
</tr>
<tr>
<td>B.3 Testing</td>
<td>67</td>
</tr>
<tr>
<td>C Geodesic Documentation</td>
<td>69</td>
</tr>
<tr>
<td>C.1 How to obtain code</td>
<td>69</td>
</tr>
<tr>
<td>C.2 Installation and Setup Guide</td>
<td>69</td>
</tr>
<tr>
<td>D Geodesic Walk-Through</td>
<td>72</td>
</tr>
<tr>
<td>E Project Support</td>
<td>81</td>
</tr>
<tr>
<td>E.1 Web site</td>
<td>81</td>
</tr>
<tr>
<td>E.2 License</td>
<td>81</td>
</tr>
<tr>
<td>E.3 CVS server</td>
<td>81</td>
</tr>
<tr>
<td>E.4 Mailing List</td>
<td>82</td>
</tr>
</tbody>
</table>
F Third Party Components and Technologies

F.1 Specification ........................................... 83
F.2 Server .................................................. 84
F.3 Client .................................................. 85
Chapter 1

Introduction

1.1 Annotation

Annotation is not a task specific to the biological community. Feedback, critiques, and exchanges are an integral part of any communication process. The original hypertext specification [8] included mechanisms for active feedback, allowing browsers to annotate by means of annotation servers. The original Mosaic browser contained an "Annotate" feature which allowed the user to save comments about a particular page to their local disk. The goal of the WWW designers was to facilitate active two-way communication rather than passive surfing, but the full specification was not widely adopted.

Another group interested in annotation is the digital libraries community. Libraries exist to serve the research needs of their constituents. Digital libraries aim to foster informal collaborations and communication through fluid and transient materials, including annotations.

The collective experience with traditional libraries has created an atmosphere directed toward slowly changing materials. Yet it should be noted that "nothing in the nature of digital technology mandates that a digital library should include only rarely changing, long-lasting documents" [43]. In general, document management systems do not deal well with versions or custom documents. Yet annotations, as a communication device, are expected to be relatively transient. Versioning of both the
reference and the annotation is critical in fluid environments to avoid skew, when the annotation refers to a different version of the reference document than currently available.

1.2 Sequence Annotation

Annotation can greatly enhance the biological value of sequence. Initially, annotation was primarily done by hand, a tedious but extremely valuable resource. A variety of systems quickly developed for automatic annotation including Genotator [30], PEDANT [23], the Institute for Genomic Research (TIGR) annotation system [37], MAGPIE [24], BEAUTY [75], and GeneQuiz [59]. Yet, concern over the quality of data, particularly annotations, has been expressed by many [46, 33, 41, 56]. These concerns prompted Wheelan and Boguski [73] to suggest that annotation should not be stored in the database but rather should be calculated on demand.

This solution has proven to be unsatisfactory. Even in the best automated environments, high quality data requires human intervention in the annotation process. Experimental results are a critical component of annotation, an aspect overlooked by Wheelan and Boguski. Researchers “want communication at least as much as they want information access” [43].

In addition to the systematic annotations that the originating lab attaches to the sequence, a wealth of information can be added by other labs: functional data derived from biochemical or genetic analysis, commentary derived from novel analytic software, errata, and even outright disagreements. To maintain quality, databases need to be heavily curated to reflect new developments, additional knowledge, and continuing research efforts. The difficulty of maintaining high quality centralized annotation is exacerbated as more laboratories contribute to the annotation process. In the short history of sequence databases, different approaches to collecting and disseminating annotation have arisen.
1.2.1 Repositories

In 1969, Margaret Dayhoff collected all of the known sequences, approximately 300 proteins and 16 nucleic acid sequences, into a book known as Dayhoff's Atlas. By 1974 the nucleic acid portion was separately published as the Handbook of Nucleic Acid Sequences. This slim book, containing mostly tRNAs, 5S RNA sequences, and an assortment of phages and viral sequences [5], was one of the first databases of genetic sequence. With advances in sequencing techniques, the Handbook had grown into a two volume set [18] by 1981.

Designed to be "the" nucleic acid sequence database, GenBank officially began at Los Alamos in 1982. GenBank’s philosophy reflected the centralized computing model dominant at the time [62]. Data is stored as simple flat files. Each database entry is owned by its submitter. Changes and annotations are therefore the responsibility of the individual or laboratory who submitted the sequence. GenBank’s size and use exploded when journals began requiring GenBank submission numbers for publications involving sequence data.

GenBank’s use is limited by a lack in uniformity between submissions. The relative quality of each entry is determined largely by the author's technical sophistication and commitment. Modern computational laboratories generate meaningful annotations across a wide variety of sequences easily. Once annotations are generated, each sequence’s owner must be contacted to request a modification to their data entry, a frustrating and arduous process.

Other large comprehensive databases include the Genome Sequence Database (GSDB) [61], EMBL [68], the Integrated Genomic Database (IGD) [58], the Protein Data Bank (PDB) [9], GenoTopographer [14], and Genome Database (GDB) [42]. While these databases are often excellent in their primary role as canonical repositories of nucleotide and protein sequence, they are less adept at dealing with annotations.

1.2.2 Curatorial Databases

With the rise of personal computing came the development of smaller databases to address the need for specialization within a particular
community. These “boutique” databases include the organism-specific databases A C. elegans Database (ACeDB) [16], Online Mendelian Inheritance in Man (OMIM) [57], FlyBase [11], Saccharomyces genome database (SGD) [10], CyanoBase [51], and the specialized databases Protein Domains (ProDom) [13], Protein Families (Pfam) [6], and the Ribosomal Database Project (RDP) [47].

A C. elegans database (ACeDB) is one particularly successful community database [16]. It has served as the central database of phenotyping, bibliographic, mapping, and sequencing information for the Caenorhabditis elegans community since 1990 [71]. ACeDB was designed with the personal computer in mind. Users are encouraged to download periodic updates from a centralized database center. Laboratories can easily add annotations to their local copy of the database. Individuals are encouraged to submit annotations to the central database curatorial group. The group then reviews the request and decides what and how it is to be incorporated into the next “official” release. Unfortunately, even in tight-knit communities, many useful annotations maintained in individual’s laboratories never get incorporated into the official release. In addition, valuable discussion, including dissension, is frequently lost in the curatorial process.

Recently, the pace of genomic sequencing has outstripped the ability of sequencing centers to annotate the sequence prior to submitting it to an archival database. Consequently a new method of annotating large genomes has arisen, the annotation party. Parties gather together a large number of experts for the express purpose of producing the best annotations possible in a limited time frame. The logistics required to organize and manage an annotation party are tremendous. A central group decides which experts to invite as participants. They must also collate the resulting annotations produced. Consequently, these are typically one time events held as a prelude to a significant publication. Two well known annotation parties were Celera’s Fly Jamboree and the Human Genome Project Consortium’s Analysis Group.

With the rise of computational biology, high throughput annotation is now possible within many laboratories. These laboratories can often annotate entire genomes quickly and efficiently. In an effort to provide to the community their results as quickly and accurately as possible, they may
produce new specialized databases. Consequently, the number of heterogenous databases containing sequence annotations is constantly growing.

1.2.3 Networking

In 1989, Tim Berners-Lee at CERN proposed the development of the World Wide Web (WWW) [8]. A number of features have made the WWW successful. Web browsers are available on all platforms. Web pages are written in a simple standardized language, the hypertext mark up language (HTML). This language can be produced in a text editor or by automated programs. A simple addressing scheme, uniform resource locators (URLs), identifies useful servers. Search engines and hot lists support the rapid location of relevant information. Cascading style sheets provide the web site with the power to easily define a uniform style for its pages [53].

Since 1995 a number of additions and expansions to ACeDB have improved its network accessibility. WebAce [72] and Ace.pm (Steve Rozen, unpublished) provide richer scripting interaction with the ACE data. More recently, Lincoln Stein has developed two complementary libraries, JADE [66] and AcePerl [67]. They utilize the power of ACE servers to provide direct access to ACE objects from Java and Perl, respectively. Many database-specific details are hidden by these libraries, providing a much needed level of abstraction. AceBrowser [65], a set of common gateway interface (CGI) scripts, was developed by Lincoln Stein from AcePerl. It provides some of ACeDB’s graphical functionality though the WWW. Most of the biologically relevant databases provide web interfaces. The result is that users now utilize web browsers to access large up-to-date back end databases.

Most databases also provide links to other relevant databases. For example, Proteome Inc.'s WormPD [32] provides links to Wormbase, GenBank, the Protein Information Resource (PIR) [4], and Swiss-Prot [3] when available. To visualize the annotations contained in these cross-linked systems requires the user to manually integrate the available information. This limitation creates a barrier to efficient information dissemination and usage [34].

An alternative method to connecting databases by web links is to
organize them into database federations or data warehouses. Database federations and data warehouses are similar methods which entail developing a global schema, the basic underlying structure, of the component databases. The principle difference between a federation and a warehouse is the way they integrate data. A federation uses an approach known as lazy integration, where data are retrieved, processed, and returned after each query. In contrast, a warehouse uses an eager approach, retrieving and integrating data in advance of a query. The tradeoff between database federations and warehousing is one of query performance versus data “freshness”.

Even if their schemas are not similar, databases can achieve some integration by the utilization of a global controlled vocabulary. One system in molecular biology is the Sequence Retrieval System (SRS) [19]. SRS deals with semistructured ASCII text by parsing and indexing mechanisms which require a controlled vocabulary. In practice, social and technical difficulties complicate the construction and maintenance of systems requiring global coordination [74].

1.2.4 Visualization

Visual presentation and manipulation of data is a central theme in biology. Much is said about automated annotation of sequence data and the building of databases, but in the end much if not most of the useful results derived from sequence data are the result of informative visualizations [60]. It is the acquired understanding of a sequence that may guide many months of experimental work. Quality visualization tools must be readily accessible to the molecular and cellular biologist who may be untrained in computer technology [36]. A variety of annotation visualization tools exist including Entrez [52], Chromoscope [7], Genome Topographer [14], Anubis [50], and Apollo [27].

The ACeDB database provides a variety of powerful visualizations. ACeDB allows positional queries through its graphical interface. By smooth transitions between relevant graphical views of the data, annotations are displayed at their appropriate magnification. For these reasons, it has been adopted by a large number of organism-specific communities. (A complete list is available at

In ACeDB’s sequence viewer the data is displayed in vertical columns. Figure 1.1 shows an example of a typical ACeDB sequence display. The axis defines the current coordinate system. Annotations are placed to each side of the axis, with forward strand information to the right and reverse strand information to the left. Each column of the display represents a different type of information, a “layer”. On the far left are columns showing this segment’s correspondence to the genetic map and overlapping contigs.

The graphical displays developed more recently for the Human Genome Project, such as Jim Kent’s Human Genome Browser [35] (Figure 1.2) and Ensembl’s Contig View [17] (Figure 1.3) have a striking similarity to ACeDB. Both the Human Genome Browser and Ensembl’s Contig View are horizontal displays, but use the same layered approach to presenting annotations.

1.3 The Distributed Annotation System

Ideally, an annotation system should give individual experts the ability to contribute to the collective annotation in a quick, robust, and mostly painless fashion. They should have complete control over their annotations in order to keep them current and relevant. Simultaneously, it should be easy for a user to obtain and visualize the most recent data about a particular region of interest. Unfortunately, these two goals are seemingly at odds in the modern sequence annotation environment.

The principle weakness of current database connection schemes is that they require strong global coordination to produce nice visualizations. The Distributed Annotation System (DAS) is an attempt to produce a system that can automatically integrate data from a number of heterogenous distributed databases while requiring only minimal global coordination. At a fundamental level, all annotations can be reduced to their coordinates relative to a particular sequence and a suggestion of how it should be drawn within a graphical display.

DAS is a client-server system where a user (the client) selects a single reference (genome) server and any number of annotation servers, as shown in Figure 1.4. DAS client software then combines the information returned
Figure 1.1: A screen-shot of ACeDB's sequence display showing the clone ZK154. Information is presented as basic colored boxes in a vertical format (as columns).
Figure 1.2: A screen-shot of Jim Kent's web based Human Genome Browser showing a portion of Chromosome 16. Information is presented as simple boxes, lines, and arrows in a horizontal orientation (as rows).
Figure 1.3: A screen-shot of Ensembl’s web browser based Contig View looking at a segment of Chromosome 6. Features are shown as colored boxes and lines. Each row, represents a different type of information.
from the servers into a graphical display. The display uses the familiar "layers" of other database visualization tools to represent the data returned from each server. A user can easily compare the basic information from multiple data sources within the region of interest. A particular annotation can then be queried to retrieve more information from its providing server. The principle change to the client-server database tools, such as those provided by ACeDB, is the concept of multiple databases systematically integrated into a single view.

![Diagram](image)

Figure 1.4: An example of the basic distributed annotation system architecture. One server is the designated reference server, in this case the Washington University Genome Sequencing Center. One or more annotation servers, shown above as Ensembl, Whitehead, and the Sean Eddy Laboratory, provide annotations relative to the reference sequence. The client, at Cold Spring Harbor Laboratory in our example, fetches data from multiple servers and automatically generates an integrated view.
1.4 Project Collaborators

This report describes the Distributed Annotation System (DAS) project, an attempt at improving annotation collection and dissemination. The project was a joint collaboration between Lincoln Stein at Cold Spring Harbor Laboratory and myself. Lincoln wrote the initial DAS outline in the Fall of 1999. Together we refined that outline into the first version of the specification in January of 2000. Lincoln then wrote a DAS server while I designed and developed a graphical client interface, Geodesic. Geodesic was written with the assistance of Rodney M. Jokerst, an undergraduate programmer. This report contains text from other DAS documents. These segments will be clearly identified. The remainder of the text is my own.
Chapter 2

System Design

The basic system is composed of a reference server, one or more annotation servers, and an annotation viewer. The reference server is responsible for serving genome maps, sequences, and information related to the sequencing process. Annotation servers are responsible for responding to requests on a region and delivering annotation. The client, an annotation viewer, is an application whose behavior is analogous to a web browser. The viewer communicates with the genome and annotation servers using a well defined language specification. Annotations are displayed within the viewer by dynamically associating a graphical representation with an annotation type, as defined by the annotation provider.

2.1 Reference Sequence

The distributed annotation system relies on there being a common "reference sequence" on which to base annotations. The reference sequence consists of a set of "entry points" into the sequence, and the lengths of each entry point. Entry points will vary from genome to genome. For some genome projects, entry points correspond to entire chromosomes. For others, entry points may be a series of contigs. In general, an entry point is a landmark within a genome to which annotations can be attached.

It is possible for each entry point to have substructure, basically a series of subsequences and their start and end points. This structure is recursive.
Annotations are a statement about a region of the reference sequence. Each annotation is unambiguously located by providing its position, start and stop, relative to a “reference sequence.” The reference sequence can be one of the entry points, or any of the subsequences within an entry point.

To give a concrete example, the *C. elegans* reference map consists of six top level entry points, one per chromosome. Each chromosome is formed from several contigs called “superlinks,” and each superlink contains one or more smaller contigs called “links.” Links in turn are composed of one or more fully-sequenced clones [12]. One could refer to an annotation by specifying its start and stop position in clone, link, superlink, or chromosome coordinates. The distributed annotation system automatically converts any coordinate system into any other.

The reference sequence server is responsible for providing the reference sequence map and the underlying DNA. The server can provide a list of sequence entry points, and given an entry point it can return its substructure. The reference server can provide arbitrarily long stretches of DNA given a reference subsequence, start position, and stop position.

### 2.2 Annotation Servers

Annotation servers are specialized for returning lists of annotations across a certain region of the genome. Each annotation is anchored to the genome map by way of a start and stop position relative to one of the reference subsequences. Annotations have an identifier that is unique to the server and a structured description of its nature and attributes. The general description of an annotation follows loosely the general feature format (GFF) (see Appendix F.1 for more information on the GFF protocol). GFF intentionally aims for a basic lowest common denominator description [15]. Annotations may also be associated with URLs where additional human readable information about the annotation is located.

Annotations have *categories*, *types*, and *methods*. The annotation *type* corresponds to a biologically significance description. Examples of annotation types may include “exon”, “intron”, “CDS” and “splice3.” The annotation *method* is intended to describe how the annotated feature was discovered, and may include a reference to a software program. The
annotation category is a broad functional category. "Homology", "variation" and "transcribed" are example categories. This structure allows researchers to add new annotation types if the existing list is inadequate without entirely losing all semantic value. The annotator is free to describe his annotations using any terms which he feels are appropriate, as DAS does not impose a controlled vocabulary. It is intended that larger annotation servers provide URLs to human-readable information that describes its types, methods and categories in more detail.

Another optional feature of annotation servers is the ability to provide hints to clients on how the annotations should be rendered visually. This is done by returning a DAS "stylesheet." Stylesheets use the type and category information to associate each annotation with a particular graphical representation, a glyph. For details on the specifics of DAS stylesheets, see Appendix A.

Although the servers are conceptually divided between reference servers and annotation servers, there is in fact no key difference between them. A single server can provide both reference sequence information and annotation information. The main functional difference is that the reference sequence server is required to serve the DNA itself, while annotation servers have no such requirement. All servers are capable of translating from one coordinate system to another.

2.3 DAS XML Specification

The main component of DAS is the specification, which defines all valid DAS communication. The DAS specification hopes to mimic HTML in features and power. The original HTML specification described a language which was human readable, easily parsed, and extensible. By using the extensible mark-up language (XML) as the return format for most queries, these features are retained in DAS. While a client can query multiple servers simultaneously, the communication between the client and any single server follows a simple client server model.

The DAS is web-based. Clients query the reference and annotation servers by sending a formatted URL request to the server. This request must follow the conventions of the HTTP/1.0 protocol. Each URL has a
Figure 2.1: The communication between the client and server is outlined in the DAS specification. Queries are URL requests and responses are generally XML, but HTML is also returned for the link query.

A site-specific prefix, followed by a standardized path and query string. The standardized path begins with the string /das. This is followed by URL components containing the data source name and a command. For example:

http://stein.cshl.org/das/elegans/features?ref=ZK154;start=1000;stop=2000

In this case, the site-specific prefix is http://stein.cshl.org/. The request begins with the standardized path, /das, and the data source, in this case /elegans. This is followed by the command /features, which requests a list of features relative to a given set of named arguments (?ref=ZK154;start=1000;stop=2000). The data source component allows a single server to provide information on several genomes.

Servers process the request and return a response as defined by the DAS specification, typically a formatted XML document. The response from the server to the client consists of a standard HTTP header with DAS status information followed optionally by an XML file that contains the answer to the query. The DAS status portion of the header consists of two lines. The first is X-DAS-Version and gives the current protocol version number, currently DAS/0.99. The second line is X-DAS-Status and contains a three digit status code which indicates the outcome of the request. The defined status codes are listed in Table 2.1.

An example HTTP header: (provided by server)

HTTP/1.1 200 OK
Date: Sun, 12 Mar 2000 16:13:51 GMT
Server: Apache/1.3.6 (Unix) mod_perl/1.19
Last-Modified: Fri, 18 Feb 2000 20:57:52 GMT
Connection: close
Content-Type: text/plain
Server Status Codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>200</td>
<td>OK, data follows</td>
</tr>
<tr>
<td>400</td>
<td>Bad command (command not recognized)</td>
</tr>
<tr>
<td>401</td>
<td>Bad data source (data source unknown)</td>
</tr>
<tr>
<td>402</td>
<td>Bad command arguments (arguments invalid)</td>
</tr>
<tr>
<td>403</td>
<td>Bad reference object (reference sequence unknown)</td>
</tr>
<tr>
<td>404</td>
<td>Bad stylesheet (requested stylesheet unknown)</td>
</tr>
<tr>
<td>405</td>
<td>Coordinate error (out of bounds/invalid)</td>
</tr>
<tr>
<td>500</td>
<td>Server error, not otherwise specified</td>
</tr>
<tr>
<td>501</td>
<td>Unimplemented feature</td>
</tr>
</tbody>
</table>

Table 2.1: Server status codes are modeled after the familiar status codes of the HTTP 1.0 protocol.

X-DAS-Version: DAS/0.99
X-DAS-Status: 200
DATA FOLLOWS ...

The valid queries are briefly summarized in Table 2.2. For all but one query, the server returns an XML document. XML languages utilize Document Type Definitions (DTDs) to specify the rules for the structure of a document. Validating parsers check the return document against the rules defined within the DTD. It is anticipated that the DTDs for the DAS specification will be standardized and made available through the project web site.

The specification outlines eight basic queries which a client can use to interrogate a DAS server. Two queries, “dsn” and “entry_points”, essentially provide information to the client about the structure of the server and the reference sequence. The “dna” query can be used to fetch a segment of DNA from a reference server. A client can request annotations, “features”, or a summary of the annotations available, “types”, from any DAS server. The “resolve” query is designed as a mechanism for translating the coordinates of one segment into the coordinate system of another segment. The servers provide a “stylesheet” to suggest representations to the client’s graphical display. When more information is desired about a particular annotation, the client makes a “link” request. The return
## Queries Summary

<table>
<thead>
<tr>
<th>Command</th>
<th>Basic Format</th>
<th>Scope</th>
<th>Returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsn</td>
<td>PREFIX/das/dsn</td>
<td>both</td>
<td>XML</td>
</tr>
<tr>
<td>entry_points</td>
<td>PREFIX/das/DSN/entry_points</td>
<td>both</td>
<td>XML</td>
</tr>
<tr>
<td>dna</td>
<td>PREFIX/das/DSN/dna?ref=REF</td>
<td>sequence</td>
<td>XML</td>
</tr>
<tr>
<td>types</td>
<td>PREFIX/das/DSN/types?ref=REF</td>
<td>annotation</td>
<td>XML</td>
</tr>
<tr>
<td>features</td>
<td>PREFIX/das/DSN/features?ref=REF</td>
<td>both</td>
<td>XML</td>
</tr>
<tr>
<td>resolve</td>
<td>PREFIX/das/DSN/resolve?</td>
<td>both</td>
<td>XML</td>
</tr>
<tr>
<td></td>
<td>segment=segmentID;ref=REF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>link</td>
<td>PREFIX/das/DSN/link?field=TAG;id=ID</td>
<td>annotation</td>
<td>HTML</td>
</tr>
<tr>
<td>stylesheet</td>
<td>PREFIX/das/DSN/stylesheet</td>
<td>annotation</td>
<td>XML</td>
</tr>
</tbody>
</table>

Table 2.2: Queries Summary

document from the “link” query is in HTML. It is anticipated that DAS clients will hand off these return documents to web browsers. For complete details on these queries, see Appendix A.

The main annotation content query, “features”, basically follows the general feature format (GFF). This helps to minimize the data sharing necessary to produce meaningful visualizations, an important consideration with a network based system.

The specification has evolved steadily since its initial web issue in March 2000. See Appendix A for the full text of the specification. See Appendix F.1 for information on HTTP, XML, HTML, and GFF.
Chapter 3
Implementations

To test and refine the distributed annotation system specification, a number of prototype systems were constructed. Figure 3.1 shows a coarse timeline of these prototypes in relation to the specification. A brief overview of their development is presented here.

![Timeline of prototype development](image)

Prototype I

Specification

Server

Prototype II

Prototype III

Figure 3.1: Timeline of prototype development. Arrows indicate continued development. After an initial development period, the servers were changed only as necessary to keep up with the specification, indicated by the dotted line.
3.1 Prototype I

The first distributed annotation system prototype pre-dates the
development of the first DAS XML specification. The prototype consists of
a cgi script written to retrieve data from multiple ACeDB databases and
combine the results into a single web page, as shown in Figure 3.2.

![Diagram of Prototype I's component architecture.](image)

**Figure 3.2:** Prototype I's component architecture. Cylinders represent different ACeDB databases. Circles represent instances of the gifaceserver. The cgi script ran on an Apache server and produced HTML documents. A standard web browser, such as Netscape, was used to view the cgi output.

The gifaceserver, a standard component of the ACeDB distribution,
was used as the annotation and sequence servers. The prototype system
used sequence from a copy of the Washington University Genome
Sequencing Center's (GSC) version of ACeDB. The initial third party
annotation databases were built using Todd Lowe's tRNAAscan-SE [45], the
GSC's SNP data [2], and the GSC's *Caenorhabditis briggsae* to
*Caenorhabditis elegans* homology data [1]. These databases were built in
October 1999 using the data available at that time.

The prototype scripts consisted of two new cgi scripts which were
added to AceBrowser, a set of cgi scripts that utilize AcePerl to converse
with ACeDB servers. The first of these scripts, select_dbs, allowed the
user to select third party data sources from a list provided by the reference
database. A cookie is then set to reflect the selections. The second script,
table2, uses the cookie information to gather the appropriate information
and display a text-based table. The contents of the four databases, one
Figure 3.3: Screen-shot of AceBrowser prototype. The top table is the information returned by queries on the reference database. The bottom table is the data returned from queries across three annotation servers, tRNAdb, SNPdb, and briggdb. Unresolved problems are seen in the error in the calculation of chromosome size and the last 6 of scores for tRNAscan-SE. In addition, the duplication of features, seen in the tRNAs above, can be attributed to an artifact of ACeDB.
reference and three annotation, were combined and displayed in a tabular form as shown in Figure 3.3.

For a number of reasons, this system was not generalizable. First, communication between AceBrowser and ACeDB depends on the structure of the underlying databases. The cgi scripts need to know detailed information about each component database a priori. In addition, the reference server was required to keep track of all third party databases available. However, this prototype was an important proof of concept and helped to identify key issues in the development of a database independent specification.

3.2 Servers

In the DAS model, a server refers to the data holding entity of the system. The server is expected to respond to the DAS specification’s defined queries with the appropriate content, usually XML. The details of server implementation are left to the various server providers. Currently existing databases can be adapted to DAS by writing a translational script as shown in Figure 3.4. This is similar to writing the cgi scripts which most databases currently utilize for their web sites. The critical difference being that cgi scripts produce HTML output whereas a translational script produces DAS XML.

![Diagram of Database, Translator, and WWW]

Figure 3.4: The architecture of the generic DAS translator script. The output format from the translator script is DAS XML.

Wormbase Server

The first translational script to be constructed was written by Lincoln Stein for Wormbase. He used AcePerl to tap into Wormbase’s ACeDB
database and retrieve the information necessary. From this information, the XML replies required by the DAS specification are produced, as shown in Figure 3.5.

![Diagram showing the components used for development of DAS specification compliant servers for wormbase and test databases.](image)

**Figure 3.5:** The components used for development of DAS specification compliant servers for wormbase and test databases.

The goal of this first translational script was simply to test the specification's feasibility with a full scale pre-existing database, Wormbase. The Wormbase data is available as a DAS reference server at http://www.wormbase.org/das/ and the script is available within the CVS repository (see Appendix E.3) at /das/server/das.

### Test Servers

Because the annotation servers produced for Prototype I utilized models similar to Wormbase, the translational script written for Wormbase was easily adapted to them. A set of test servers was established at http://128.252.205.72:8081/cgi-bin/das/ (now available at http://wolfie.wustl.edu/cgi-bin/das/) using the third party databases developed for Prototype I. The reference server was reduced to contain only the sequence and map information from Wormbase. In addition, a fourth third party database was built from the Wormbase annotations for exons and introns [76].

To keep pace with a rapidly evolving specification, the server script was modified frequently by both Lincoln and myself. For example, the original Wormbase server did not implement the stylesheet query but the test servers were modified to return a hand-made flat file in response to the stylesheet query. In collaboration with Robert Citek in the Michael Brent Lab, a brief HOWTO document was written for the ACeDB based servers (see Appendix B).
Validator

To test early server implementations, Rodney M. Jokerst wrote a DAS specification validator in May 2000. In the spirit of XML validating parsers, the validator checks the XML return document from a query against an XML data type definition (DTD). The validator is not a true XML validator, in that it is not capable of validating any XML document against an arbitrary DTD. Even so, this simple tool permitted an easy way to test the contents of early server implementations against the expectations of DAS clients.

3.3 Prototype II

Once completed, the test servers were used in the development of a graphical client interface called Geodesic. The test servers and Geodesic version 0.2, the first public release, together constitute the DAS Prototype II system, as shown in Figure 3.6. This system was presented at both the Bioinformatics Open Source Conference (BOSC) and Intelligent Systems in Molecular Biology (ISMB) Conference in August 2000. This was the first DAS prototype system utilizing the XML communication specification.

Design Decisions

A number of early design decisions directed the development of the first DAS client program. Because the client is a graphical interface, the capabilities of the programming language to provide a mostly platform independent interface is critical. A number of graphical toolkits for a variety of languages were examined. Most were found at that time, November 1999, to be lacking. In the end, Java was selected as the development language.

To maximize portability, the v0.2 version of Geodesic was developed using Java 1.1.3, as Macintosh JVMs are only 1.1 compatible. This limited the interface to using only AWT components. The Neomorphic Genome Software Development Kit (NGSDK) provides powerful mapping and graphical display capabilities in Java's AWT componentry. Consequently, it
was utilized for the rapid production of graphical map displays.

The return documents for DAS are XML compliant. This permits the utilization of pre-existing XML parsers within Geodesic. XML parsers are roughly divided into two categories, event-driven and data-object model (DOM). DOM parsers create a tree data object from the XML structure. These tree objects require a high memory overhead and can be inefficient for certain types of data accesses. As genomic data could be quite large, this could be problematic. Event driven parsers fire a particular event type for each encountered element. They tend to be smaller and faster, but handling the event queue can result in complicated code. For Geodesic, the freely available event-driven AlItred parser was utilized (see Appendix F.3).

**Geodesic v0.2**

The resulting client, Geodesic v0.2, contained most of the core functionality of a valid DAS client. This version focused on producing a working graphical prototype as quickly as possible. The general structure of
the code is shown in Figure 3.7. XML return documents are parsed by the \texttt{Ælfred} parser into the Geodesic datastructures by the \texttt{DasXmlHandler}. Geodesic then uses the datastructures to interface with the Neomorphic Genome Software Development Toolkit (NGSDK) to produce graphical maps. Figure 3.8 shows a screen-shot of Geodesic v0.2 as presented at BOSC 2000.

![Diagram](image)

Figure 3.7: The basic structure of Geodesic v0.2.

The NGSDK was designed to permit a number of different graphical constructions common to biology including annotation maps, sequence assemblies, sequence quality plots, and sequencing traces. In Geodesic v0.2, the NGSDK was utilized for the graphical map displays within the main interface (as seen in Figure 3.8) and the entry points display (as seen in Figure 3.9). To produce these graphical displays, only the general annotation map component of the NGSDK, NeoMap, was utilized.

Specializing NeoMap was awkward and resulted in slow and memory intensive displays. While the NGSDK provides a rich set of graphical capabilities, it was not able to represent the complete set of glyphs outlined in the DAS specification. In addition, the NGSDK provides a large number of capabilities not utilized in Geodesic. Because these unused elements cannot be separated from NeoMap, they unnecessarily increase the relative size of the Geodesic release. Lastly, Neomorphic was acquired by Affymetrix late in 2000. This acquisition leaves the long term availability and maintenance of the toolkit in question. These problems prompted the decision, in August 2000, to eliminate the NGSDK from subsequent developments.

In addition, Geodesic v0.2 had other limitations. It used a simple network management strategy. Sources were queried in a sequential fashion
Figure 3.8: Screen-shot of Geodesic v0.2 in action. The current display is looking at the *C. elegans* clone F18C5. The currently selected annotation layers include the tRNA layer, at top showing red boxes, and the ORF database, second layer showing blue boxes.
Figure 3.9: Screen-shot of Geodesic v0.2's Entry Points dialog box (rotated 90 degrees). The entry points are the hierarchical components of the genome's map. The entry points dialog gives a graphical display of the genome's structure for browsing and traversing the data contained therein.
creating a slow system that was highly susceptible to network failure. In addition, the use of only pure AWT interface componentry resulted in a rather clunky interface. But even with its problems, the Prototype II system was successful at generating positive discussion at both the BOSC '00 and ISMB '00 meetings. Its development also resulted in a number of modifications to the specification.

3.4 Prototype III

The same system components shown in Figure 3.6 were utilized for the development of the Prototype III system. The major change is the complete rewrite of Geodesic. The specification and servers have largely stabilized, with only minor changes from prototype I. The improved version of Geodesic eliminates dependencies on the NGSDK, introduces threading into its network management, and moves interface components to Swing (Java 1.2). A detailed discussion of the current Geodesic software is the subject of Chapter 4.
Chapter 4

Geodesic

The current Geodesic code base is roughly divided into two major components. The first component, known as Hydra, handles the functionality of a DAS client without any specific interface (graphical or text based). The other component manages the graphical interface for Geodesic. Every effort was made to keep the two separate. The result is shown in Figure 4.1.

The description presented here refers to the current development code. When a path is given, it refers to the object's location within the CVS's geodesic directory. See Appendix E.3 for information on how to obtain a copy of Geodesic. The current status report and TODO lists are available through the programmer's notes section of the project web site (http://biodas.org/geodesic/notes.html). The Geodesic code base is also documented with javadoc.

4.1 Hydra

Hydra manages the network queries, handles the event queue spawned by Ælfred, builds the appropriate data structures in response to a query, and does a number of data checks. Essentially, it is an attempt at a threaded "interface-free" client package. The basic architecture of Hydra is shown in Figure 4.2.
Figure 4.1: A screen-shot of the current development version of Geodesic. The view is on clone ZK154 using sources from the *C. elegans* test server set.
Figure 4.2: Hydra’s basic architecture. The network queries are handled by HydraSourceManager threads. Threads persist until a source is removed from the interface. The threads are collectively managed by the HydraManager.

Ælfred

Incoming XML is parsed by a slightly modified Ælfred parser (/hydra/aeelfred/). Ælfred is a light, DTD aware, event-driven parser. To speed up development of the client, it was modified to be “DTD unaware”. As Ælfred encounters XML components it triggers event calls to registered handler. The DasXmlHandler (/hydra/handler/) responds to these events by building the appropriate data structure (/hydra/datastruct/) for the query. The handler uses a single stack to store incoming information. When an end tag is encountered, the stack is popped back to the start tag and the appropriate data structure element is built.

Datastructures

The datastructures are responsible for holding the information returned from a query and for performing data conversions and checks. Each DAS query has a datastructure associated with it. Most datastructures are composed of one or more DasObjects, which serves as the underlying basic storage object.
Managers

Network management is fully threaded. Each data source is a separate thread. Each thread is designed to listen and process queries. For each query the thread determines the steps necessary to complete the request. In most cases, this involves spawning a new thread to send a properly formatted URL query through the handler to the server. The handler then returns the appropriate datastructure and the handler thread dies. This two-tiered handling of threading allows for queries to be killed before they have returned.

The Manager collects the data source threads, SourceManagers, providing a single interface for distributing request to all of the threads. Source threads are created the first time a data source is queried and only destroyed when a data source is removed from the current view.

Geodesic interfaces with Hydra by extending the manager’s functionality. The primary extension permits the HydraSourceManagers to place the appropriate graphics component in the Swing EventQueue. This permits the data source threads to interact with the graphical interface.

4.2 Graphical User Interface

The Swing counterparts to the standard AWT components provide a much sharper look and feel and more consistent behavior. Consequently, Swing components are used whenever possible.

Preferences Files

Geodesic generates two storage files, preferences.xml and bookmarks.xml. The preferences file holds user-specified information about the interface. The bookmarks file allows Geodesic to remember information about previously visited servers. With the parser-to-datastructure elements already in place within Hydra, these files are also kept as XML and have corresponding datastructures.
Stylesheets

Stylesheets are kept as large hash tables, known as the StyleBank. When features are to be displayed, their type and category are hashed to the StyleBank to identify what glyph type is desired (/das/datastruct/BankEntry). The appropriate glyph is then allocated and associated with the stylesheets' attributes (/das/datastruct/AttributeList) and feature (/das/datastruct/Feature). At the lowest level, Geodesic uses a programmatic default of a blue box for all features. This programmatic stylesheet is overridden in each data source by the style bank returned from a stylesheet query, if available.

Sequence Maps

The main display is the set of graphical maps that display the contents of a features request across multiple sources. For version 0.2, the Neomorphic Genome Software Development Kit (NGSDK) was utilized. The NGSDK write out uses JComponents to represent each possible glyph type (/gui/glyphs/) and a single JComponent as the DasMap (/gui/map/DasMap). The glyphs are drawn relative to their preferred size which is set by the map's custom layout manager (/gui/map/DasMapLayout).

The layout manager does crude collision avoidance when setting the position and size of each glyph. The collision avoidance algorithm is a simple layered approach. An array is used to store the endpoint of each layer's right most glyph. Each index of the array represents a new layer (row) within the graphical map. The vertical size of any given row is fixed. Each glyph's height, as returned by the stylesheet indicates the number of rows on which it will reside (and hence its size in pixels). The array is initialized to the maximum negative value. The glyphs, sorted by start position, are then compared with the contents of the array until a position is found where the start value of the glyph is greater than the current array contents for the number of layers required by the glyph. Once found, the endpoint of the glyph is stored in the array in the glyph's layers. In this way, graphics collisions are avoided.

Maps are placed into a containment hierarchy for display within the
MainWindow. For annotation sources, maps are packaged within a JPanel to create a PackagedMap. PackagedMaps are responsible for handling the labeling and menu controls specific to each instance of a map. Figure 4.3 shows the object-to-interface map for a PackagedMap. Packaged maps are placed in a DasMapContainer, which handles the layout of maps relative to each other. A DasMapContainer is then placed in a scrollpane for proper scrolling.

![Diagram of MapLabel and PackagedMap](image)

Figure 4.3: The object-to-interface map for a single packaged map.

**Interface Layouts**

Geodesic can display data in three unique layouts, shown in Figure 4.4. The layouts differ in the position of the reference map (ReferenceMap). The user can place the reference map at the top of the display, the bottom of the display, or in the center. If placed in the center, then the incoming data is split by strand, with forward strand information above and reverse strand information below the axis. If the reference map is at the top or bottom of the display then no distinction in strand is made.

To create the three distinct MainWindow layouts, the PackagedMap object is specialized into ForwardMap and ReverseMap. A forward map displays annotations from the bottom upwards. Reverse maps are top down displays. Similarly the DasMapContainer is specialized into an UpperDasMapContainer and a LowerDasMapContainer. The ReferenceMap is unique. It utilizes a ForwardDasMap, a ReverseDasMap, and a specialized Axis glyph to handle three possible layouts, the direct analogs to the MainWindow layouts. The goal is to graphically adjust the
Figure 4.4: The three basic layouts available within the Geodesic browser. All layouts are described by the position of the reference map.

layout algorithms accordingly to keep annotation glyphs moving away from the reference map's axis glyph, as when adjusting to a collision. Figure 4.5 shows the object-to-interface map of the MainWindow for the most common case, when the reference map is centered.

Dialogs and Menus

A number of dialogs and menu options are necessary to produce a useful interface. The main elements are briefly described here. For a complete walk-through with screen-shots, see Appendix D.

The most important are those that deal with source management, the Genome Selection dialog and the Annotations Selection dialog. The information presented within these dialogs comes directly from the bookmarks.xml file. Sources presented in the annotations dialog must match the reference source selected (the mapmaster tag) in order to be displayed and available for selection.

Two dialogs are available for use in selecting a location within the genome to view, the Entry Points dialog and the Region dialog. The entry points dialog provides a hierarchical presentation of the reference sequence. It is useful for browsing through the genome. If the entry point is known a priori, the region dialog allows a user to input it's name directly.

The data contained with Geodesic can be exported into a number of
Figure 4.5: The object-to-interface map for Geodesic. The UpperMapContainer and LowerMapContainer are in geodesic/gui/. The layout of the main panel is contained by the MainWindow object in the same directory. The ReferenceMap is defined in geodesic/gui/map/. The scrollpanes upperScroll and lowerScroll contain one or more packaged maps. The scrollpane refScroll contains a single ReferenceMap.

formats, including DAS XML, tab delimited GFF, and FASTA. These export options can be exercised over all sources, a single source, or a selected feature.

When a user is interested in a particular annotation, the feature details dialog is designed to present the user with the information contained within the feature's XML. From the feature details dialog the user can request more information. These requests cause Geodesic to query the server for link information. The return documents from a link query are HTML documents that ultimately should be handed off to the system's web browser. Instead, they are currently viewed within Java's JEditorPane which can display simple HTML.
Chapter 5

Comments and Future Work

A number of elements of DAS encourage annotation providers to contribute to the system. Because an annotation provider needs only the genome map from a sequence provider (reference server), the overhead of creating an annotation source is limited. In addition, the use of DAS stylesheets gives annotation providers control over not only the data content but also the representation of that data within a DAS client. The use of links to refer back to local web pages provides the annotation provider with even greater power of expression and content control.

DAS distributes data sources across the Internet, increasing robustness and improving scalability over monolithic systems. This distribution of data encourages a divide-and-conquer approach to annotation, where experts provide and maintain their own annotations. It also permits annotation providers to disagree about a particular region, encouraging informative dissension and dialogue. Large centralized databases are typically slow to respond to changes and new work. In contrast, DAS annotation providers are directly associated with the annotations they provide. This accountability encourages annotation providers to keep their information current.

Unlike database federations, which enforce a common schema, DAS specifies only the communications between client and server. To limit bandwidth requirements, each annotation is represented within DAS by a minimal, lowest common set of qualities. Users can link back to the providing database for additional information on annotations of interest.
Therefore, the details of a database and server implementation is left open to server providers. The separation of sequence and map information from annotation allows them to be stored and represented in a variety of database schema. It is anticipated that a number of database alternatives will arise.

Because a DAS browser needs to be installed locally only once users will be encouraged to use DAS over keeping local copies of the database. The user is presented with an automatically integrated perspective of data from a large number of sources. Data is kept as up to date as the servers provide. The servers translate between multiple levels of relative coordinates seamlessly, allowing a user to browser through the genome data. DAS users ultimately control the quality of the data they view by the selection of which annotation providers to view. In this way “market forces” will ensure that successful annotation layers survive while outdated layers disappear from the client’s browser. Word of mouth and publications will be driving forces in user selection.

Unfortunately, DAS does not solve all of the problems with genome annotation system. Users will be required to learn yet another system, the DAS interface. To ease this learning curve, the Geodesic interface is similar to many of the other genome displays already available. When users link back to the providing database, they can expect a number of formats for this information. Users already face this problem daily through the database web interfaces.

5.1 Specification

Resolve

The “resolve” query is problematic. It was intended as a mechanism for translating annotations between versions of the reference database. However, the implementation details of resolve are complicated. There is some confusion as to who should be responsible for resolving version skew—the client, annotation server, or reference server. In a finished genome this is less of an issue as changes to the reference are expected to be more subtle and infrequent. One possible alternative to resolve is for queries to permit
an optional version number argument. The client could then either negotiate between the reference and annotation servers to find a common version or opt to "punt", declaring the two versions unresolvable. None of the currently available DAS servers implement the resolve query. It is expected that the resolve query will not be part of the v1.0 specification.

**DTDs**

Currently, no "official" data type definitions (DTDs) are available for the specification. Many off the shelf XML parsers require the existence of DTDs. Therefore, DTDs should be written and made available at http://biodas.org/. At that time, standard XML validating parsers would be appropriate for confirming the correctness of developing servers.

**Bandwidth**

It is anticipated that for large genomes, bandwidth will be a limiting factor in DAS. One solution would be the introduction of compression into network transfers. The servers would compress the XML return documents before sending them along the network. Clients would decompress the documents as they arrive. This technique only helps if the gain in transfer time exceeds the increased overhead of compression algorithms. A number of groups have been working on the issue of efficient XML compression, including AT&T Research Labs [69] and XML Solutions [63].

Modification of the features request to specify a "zoom depth" would also reduce bandwidth requirements. For example, single nucleotide polymorphism annotations are too fine grained to be visible at the full chromosome level. The client could determine a size threshold based on the current zoom depth and request only those annotations that are within the threshold. A detailed study of the network performance of Geodesic is needed to shed some light onto the severity of bandwidth limitations.
5.2 Servers

Even with the limited expressivity of the DAS feature format (based loosely on GFF), a number of elements of a given feature table were difficult to retrofit cleanly onto the existing Wormbase database. For example, DAS distinguishes between human readable labels and database identifiers. Within Wormbase, however, a feature may simply be referred as “ZK154.3” rather than something more informative, such as “tRNA: ARG”. Usually the information desired is within Wormbase, but representation heterogeneity makes automatic conversion scripts difficult to write.

Two changes are needed to the bank of test servers to improve the demonstrations of Geodesic. First, the current servers do not respond to link queries. Modifications to the server are needed to produce working link queries which produce relevant and nice HTML pages. Secondly, the databases used in the test server cluster used Fall 1999 data from C. elegans. Since that time most of the gaps in the sequence have been closed. The databases should be updated to reflect modern data.

DAS specific ACeDB

After releasing the initial DAS prototype system, it became apparent that a DAS specific ACeDB system would be useful to improve the testing capabilities of further client development. Consequently, specialized DAS specific models were developed for ACeDB and a new translation Perl script is being written in order to effectively produce a DAS specific ACeDB package.

DAS specific models for ACeDB require a data provider to create a DAS specific database for their annotations. Consequently, it is not expected that pre-existing databases will be converted into this format. However, this may be a suitable method for new DAS annotation databases to be constructed. Ultimately, a number of supporting format interconversion scripts will be needed to simplify the input of DAS ready data into the database.
Other Servers

Experience with setting up a third party server demonstrated a number of the technical difficulties third parties can expect to experience. To minimize resistance to DAS, the server installation and setup should be made significantly easier. The HOWTO of Appendix B is the first step. The construction of more server packages utilizing other databases, such as mySQL, will give annotation providers more choices and greater flexibility. A lightweight server package, perhaps a modified Apache server, would be another alternative. In general, more alternatives means less resistance to producing DAS compliant servers.

A number of groups have posted to the DAS mailing list their intentions to produce either reference or annotation servers. (See Appendix E.4 for information on the mailing list.) These groups utilize a variety of databases, methods, and organisms to contribute to the DAS community. The Berkeley Drosophila Genome Project group has produced a prototype GadFly reference server. The BioJava community developed a servlet and basic client over existing BioJava components. The Ensembl group is developing a human reference server. Oak Ridge National Labs is working on a DAS based server. Other groups considering working on DAS servers include Incyte Genomics, Electric Genetics, and the Rice Consortium in the Philippines.

5.3 Clients

Geodesic

The movement to Java 1.2 over Java 1.1 reduces the portability of Geodesic. As Java 1.3 is currently under development for Macintosh, it is anticipated that the loss in portability will be for a short time only.

Grouping and labeling of glyphs will need to be added to the current map strategy. This can be accomplished by the creation of a glyph object which can contains other glyphs. This grouped glyph is responsible for drawing the composite of the glyphs it contains. It can then be drawn as a single glyph, assuring that the groups will remain intact.
With the enormous quantity of information possible within a finished genome, Geodesic will likely face significant memory issues. The current implementation allocates a glyph object for every feature returned by a server. For extremely large data sets, it may be more efficient to use an alternative allocation strategy. A detailed evaluation of Geodesic's memory usage is needed.

The bottleneck in Geodesic's performance is network queries. Even with threading, a network query typically creates a noticeable response delay. A number of client side strategies could be used to reduce the intrusiveness of these delays.

The easiest solution is to indicate to the user that Geodesic is busy. Changes in the interface, such as greying out a menu item, prevent users from executing methods which are not valid while Geodesic is in a transitional state. A status bar provides the user feedback with feedback as to the progress of a network request. In addition to this feedback, the client should do more intelligent fetching. If a new request is a subregion of the current dataset, no query is needed. Both of these techniques are being written into the current version of Geodesic.

A more sophisticated solution would be to have Geodesic pre fetch information which it believes will be needed soon. This requires some notion of what behavior to expect out of a user. The more accurate the prediction, the better this strategy becomes. However, it would also required more sophisticated network management capabilities. A method would be needed to determine the status of a requested query: not fetched, in progress, or already fetched. Pre fetching would need to be abandoned (threads killed) when proven incorrect. Lastly, pre fetching would consume additional memory and system resources. These complications make pre fetching beyond the scope of the current Geodesic development.

Other Clients

A Perl alternative to Geodesic is under development by Allen Day and Lincoln Stein. This server side script, called DasView, connects to one or more DAS servers and constructs an integrated image map which can be displayed in a regular web browser. As no software installation is needed by users, DasView presents an alternative for those interested in only casual
use of the DAS system.

According to reports on the DAS mailing list, other groups are also interested in developing DAS clients. The BioJava group has started development on a simple DAS client which utilizes the BioJava objects to drive a textual interface. They hope to also build a graphical package using their sequence renderer engine. Another graphical client under consideration is the adaptation of Apollo to DAS data. A text based interface to Hydra and a DAS Perl module have been mentioned as methods of providing middle-ware interfaces to DAS.
Appendix A

Specification Summary

The following is a summary of the current DAS specification, minus the problematic resolve query. It is based on v0.99 (February 2001) of the specification. The specification was originally written by Lincoln D. Stein, Robin D. Dowell, Rodney M. Jokerst, and Sean R. Eddy. The current version of the specification is available at http://biolas.org/documents/spec.html.

February 19, 2001     Version 0.99
September 6, 2000    Version 0.98
June 14, 2000        Version 0.95
March 13, 2000       Version 0.90
January 6, 2000      Version 0.85
November 20, 1999    Version 0.20

Figure A.1: The version history of the DAS specification

A.1 The Queries

This section lists the queries recognized by sequence and annotation servers. Each of these queries begins with some site-specific prefix, denoted here as PREFIX. The other meta-variable used in these examples is DSN, which is a symbolic data source name.
Data Sources  The dsn query returns a list of data sources available from the server. A single annotation provider (unique PREFIX) may present a number of annotation databases by using different data source names.

Scope: Reference and annotation servers.
Command: dsn
Format: PREFIX/das/dsn
Arguments: none
Return Document:

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASDSN SYSTEM "dasdsn.dtd">
<DASDSN>
  <DSN>
    <SOURCE id="id1"> source name 1 </SOURCE>
    <MAPMASTER> URL <MAPMASTER>
    <DESCRIPTION> descriptive text 1 </DESCRIPTION>
  </DSN>
  <DSN>
    <SOURCE id="id2"> source name 2 </SOURCE>
    <MAPMASTER> URL <MAPMASTER>
    <DESCRIPTION href="url"> descriptive text 2 </DESCRIPTION>
  </DSN>
  ...
</DASDSN>
```

<!DOCTYPE> (required; one only) The doctype indicates which formal DTD specification to use. For the dsn query, the doctype DTD is "dasdsn.dtd".

<DASDSN> (required; one only) The appropriate doctype and root tag is DASDSN.

<DSN> (required; one or more) There are one or more <DSN> tags, one for each data source. Each <DSN> contains one <SOURCE> tag, one <MAPMASTER> tag, and optionally one <DESCRIPTION> tag.

<SOURCE> (required; one per DSN tag) This tag indicates the symbolic name for a data source. The symbolic name to use for further requests can be found in the id (required) attribute. The tag body contains a human-readable label which may or may not be different from the ID.

50
<MAPMASTER> (required; one per DSN tag) This tag contains the URL (site-specific.prefix/das/data_src) that is being annotated by this data source. For an annotation server, this is the reference server which is being annotated. By definition, a reference server annotates itself and therefore would echo its own URL.

<DESCRIPTION> (optional) This tag contains additional descriptive information about the data source. If an href (optional) attribute is present, the attribute contains a link to further human-readable information about the data source, such as its home page.

Entry Points The entry_points query returns the list of sequence entry points available and their sizes in base pairs.

Scope: Reference and annotation servers.
Command: entry_points
Format: PREFIX/das/DSN/entry_points[?ref=REF]
Arguments:

ref (optional) If a sequence reference ID is provided in the ref argument, the query will return the components of the sequence (its subsequences) rather than the list of top-level entry point sequences.

type (optional) For ACEDB servers, the type parameter provides the class of the reference sequence, sequence by default.

Return Document:

<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASEP SYSTEM "dasep.dtd">
<DASEP>
  <ENTRY_POINTS href="url" version="X.XX" id="id">
    <SEGMENT id="id1" start="start1" stop="stop1">text</SEGMENT>
    <SEGMENT id="id2" start="start2" stop="stop2">text</SEGMENT>
    <SEGMENT id="id3" start="start3" stop="stop3">text</SEGMENT>
    ...
  </ENTRY_POINTS>
</DASEP>
<!DOCTYPE> (required; one only) The doctype indicates which formal DTD specification to use. For the entry_points query, the doctype DTD is “dasep.dtd”.

<DASEP> (required, one only) The appropriate doctype and root tag is DASEP.

<ENTRY_POINTS> (required, only one) There is a single <ENTRY_POINTS> tag. It has a version number (required) in the form “N.NN”. Whenever the sequence map changes, the version number should change as well.
If the entry points are not “top level”, that is, if they were generated by a request for the substructure of a sequence, then the id attribute will be present and will indicate the database ID of the reference sequence.
The href (required) attribute echoes the URL query that was used to fetch the current document.

<SEGMENT> (optional; zero or more) Each segment contains the attributes id, start, and stop (all required). The id is a unique identifier, which can be used as the reference ID in further requests to DAS. The start and stop each indicate the position of the sequence within the reference sequence. If no reference sequence is provided (i.e., the sequence is a “top level” object), then start will be 1 and stop will be the full length of the segment object. Zero and negative numbers are acceptable. If the coordinate is off the end of the reference sequence (but not off the end of the genome), the server performs the necessary join. If stop < start then the request returns the reverse complement of the segment. If no segments are provided for a given query, it is assumed to have no entry_points (substructure).
The body of the <SEGMENT> sections contains human-readable text (optional) for the purposes of display and selection.

DNA  The dna query returns the DNA corresponding to the indicated segment.
Scope: Reference servers.
Command: dna
Format: PREFIX/das/DSN/dna?ref=REF[start=x;stop=y]
Arguments:

ref  (required) The ID of a sequence landmark (an entry point or
subsequence).

**start (optional)** The start position of the segment, where 1 is the first base pair in the sequence. If this argument is provided **stop** must also be provided. If not provided, then the entire length of the entry point or subsequence is returned. Zero and negative numbers are acceptable. If the coordinate is off the end of the reference sequence (but not off the end of the genome), the server performs the necessary join.

**stop (optional)** The end position of the segment; mandatory if **start** is provided. If **stop** < **start** then the request returns the reverse complement of the segment.

**Return Document:**

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASDNA SYSTEM "dasdna.dtd">
<DASDNA>
<SEQUENCE id="id" start="start" stop="stop" version="X.XX">
<DNA>
atttcttgccgtaataagagtctctcaatgagactctcagaaagaaattgataaaatatat
ataaataaaaataatcttggtgactccgctttcatctccgagcagttttctgtcc
gaaacaatgaacactattataacatatctctcagaaatgtagatatttaagcgaaggtagtgc
acatcaaggccgctcaaacggaaaaatatttttgcaagaatca
</DNA>
</SEQUENCE>
</DASDNA>
```

**<!DOCTYPE> (required; one only)** The doctype indicates which formal DTD specification to use. For the dna query, the doctype DTD is "dasdna.dtd".

**<DASDNA> (required; one only)** The appropriate doctype and root tag is DASDNA.

**<SEQUENCE> (required; one only)** There is a single **<SEQUENCES>** tag. It has the attributes id, which indicates the reference ID for this sequence, start and stop, which indicate the position of this segment within the reference sequence, and version, which provides the sequence map version number. All four attributes are required.

53
<DNA> (required; one only) This tag surrounds the DNA data. It has the attribute length (required), which indicates the length of the DNA. The DNA is found in the body of the tag and is required. DNA will be lower-case and adhere to the IUPAC code conventions.

Summary Information  The types query returns a summary of the annotation available for a segment of sequence.

Scope: Annotation Servers
Command: types
Format: PREFIX/das/DSN/types?ref=REF
            [:start=X;stop=Y;type=TYPEPATTERN]
Arguments:

ref  (required) The ID of a sequence landmark (an entry point or subsequence).

start (optional) The start position of the segment, where 1 is the first base pair in the sequence. If this argument is provided stop must also be provided. If not provided, then the entire length of the entry point or subsequence is returned. Zero and negative numbers are acceptable. If the coordinate is off the end of the reference sequence (but not off the end of the genome), the server performs the necessary join.

stop (optional) The end position of the segment; mandatory if start is provided. If start < stop then the request returns the reverse complement of the segment.

type (optional) A GNU regular expression to be used for filtering annotations on the type field. Regular expressions follow the syntax of Version 8 regular expressions that form the core of the Perl and GNU regular expression libraries; see the UNIX manual page regex(7).

Return Document:

<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASTYPES SYSTEM "datypes.dtd">
<DASTYPES>
  <GFF version="0.98" href="$url">
  <SEGMENT id="id" start="start" stop="stop" version="X.XX" label="label">

54
Annotations The features query returns the annotations across a segment of sequence.

Scope: Reference and annotation Servers
Command: features
Format: PREFIX/das/DSN/features?ref=REF
[start=X&stop=Y;type=TYPEPATTERN]
[category=CATEGORYPATTERN;categorize=yes—no]

Arguments:

ref (required) The ID of a sequence landmark (an entry point or subsequence).

start (optional) The start position of the segment, where 1 is the first base pair in the sequence. If this argument is provided stop must also be provided. If not provided, then the entire length of the entry point or subsequence is returned. Zero and negative numbers are acceptable. If the coordinate is off the end of the reference sequence (but not off the end of the genome), the server performs the necessary join.

stop (optional) The end position of the segment; mandatory if start is provided. If start < stop then the request returns the reverse complement of the segment.

type (optional) A Version 8 regular expression to be used for filtering annotations on the type field. See the UNIX manual page regex(7).

category (optional) A Version 8 regular expression to be used for filtering annotations by the category field. See the UNIX manual page regex(7). If both type and category are provided, they are combined by a logical OR.

categorize (optional) Either "yes" or "no" (default). If "yes", then each annotation will include its functional category.

Return Document:

<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASGFF SYSTEM "dasgff.dtd">
<DASGFF>
  <GFF version="1.2" href="url">
    <SEGMENT id="id" start="start" stop="stop" version="X.XX" label="label">
      <FEATURE id="id" label="label">
        <TYPE id="id" category="category" reference="yes|no">
          type label
        </TYPE>
      </FEATURE>
    </SEGMENT>
  </GFF>
</DASGFF>
The positions of all returned annotations are given relative to the indicated reference sequence.

<!DOCTYPE> (required; one only) The doctype indicates which formal DTD specification to use. For the feature query, the doctype DTD is "dasgff.dtd".

<DAS> (required; one only) The appropriate doctype and root tag is DASGFF.

<GFF> (required; one only) There is a single <GFF> tag. Its version attribute indicates the current version of the XML form of the General Feature Format. The current version is (arbitrarily) 0.95. The href (required) attribute echoes the URL query that was used to fetch the current document.

<SEGMENT> (required; only one) There is one <SEGMENT> tag, providing information on the reference segment coordinate system. The id, start and stop attributes indicate the position of the segment. The version attribute indicates the current version of the sequence map. The id, start, stop, and version attributes are required. The optional label attribute provides a human readable label for display purposes.

<FEATURE> (required; one or more per SEGMENT) There are one or more <FEATURE> tags per <SEGMENT>, each providing information
on one annotation. The id attribute (required) is a unique identifier for the feature. It can be used as a reference point for further navigation. The label attribute (optional) is a suggested label to display for the feature. If not present, the id attribute can be used instead.

<TYPE> (required; one per FEATURE) Each feature has just one <TYPE> field, which indicates the type of the annotation. The attributes are id (optional), which is a unique id for the annotation type and can be used to retrieve further information from the annotation server (see Linking to a Feature), and the category (optional) attribute, which provides functional grouping to related types. The reference server's annotations can consist of additional overlapping landmarks (parents, children, and neighbors), which should be marked "yes" in the third attribute reference (optional, defaults to "no") to indicate that the feature is a structural landmark within the map (this feature can be annotated). The tag contents (optional) is a human readable label for display purposes.

<METHOD> (required; one per FEATURE) Each feature has one <METHOD> field, which identifies the method used to identify the feature. The id (optional) tag can be used to retrieve further information from the server. The tag contents (optional) is a human readable label.

<START>, <END> (required; one apiece per FEATURE) These tags indicate the start and end of the feature in the coordinate system of the reference sequence given in the <SEGMENT> tag. The relationship between the feature start and stop positions and the segment start and stop is that the two spans are guaranteed to overlap.

<SCORE> (required; one per FEATURE) This is a floating point number indicating the "score" of the method used to find the current feature. The number can only be understood in the context of information retrieved from the server by linking to the method. If this field is inapplicable, the contents of the tag can be replaced with a - symbol.

<ORIENTATION> (required; one per FEATURE) This tag indicates the orientation of the feature relative to the direction of transcription. It may be 0 for features that are unrelated to transcription, +, for features that are on the sense strand, and -, for features on the antisense strand.

<PULSE> (required; one per FEATURE) This tag indicates the position of the feature relative to open reading frame, if any. It may be one of the integers 0, 1 or 2, corresponding to each of the three reading frames, or - if the feature is unrelated to a reading frame.
<GROUP> (optional; if present, one per FEATURE) The <GROUP> section is an oddity, as it is derived from an overloaded field in the GFF flat file format. It provides a unique "group" ID that indicates when certain features are related to each other. The canonical example is the CDS, exons and introns of a transcribed gene, which logically belong together. The id (required) tag provides an identifier that should be used by the client to group features together visually. Unlike other IDs in this protocol, the group ID cannot be used as a database handle to retrieve further information about the group. Such information can, however, be provided within <GROUP> section, which may contain up to three optional tags.

<NOTE> (optional; if present, one per GROUP) A human-readable note in plain text format.

<Link> (optional; if present, one per GROUP) A link to a web page somewhere that provides more information about this group. The href (required) attribute provides the URL target for the link. The link text is an optional human readable label for display purposes.

<TARGET> (optional; if present; one per GROUP) The target sequence in a sequence similarity match. The id attribute provides the reference ID for the target sequence, and the start and stop attributes indicate the segment that matched across the target sequence. All three attributes are required. More information on the target can be retrieved by linking back to the annotation server. See Linking to a Feature.

Linking to a Feature The link query can be issued in order to retrieve further human-readable information about an annotation. It is best to pass this URL directly to a browser, as the type of the returned data is not specified (it will typically be an HTML file, but any MIME format is allowed).

Scope: Annotation Servers
Command:<link
Format:PREFIX/das/DSN/link?field=TAG;id=ID
Arguments:

field (required) The field to fetch further information on. Options are:

- feature – the feature itself
- type – the feature type
- method – the feature method
- category – the feature category
- target – the target, applicable to sequence similarities only

**id (required)** The ID of the indicated annotation field.

**Returns:** A web page.

**Stylesheet**  The stylesheet query can be issued to an annotation server in order to retrieve the server's recommendations on formatting annotations retrieved from it. These recommendations are not normative. A viewer is free to use any display format it chooses.

- **Scope:** Annotation Servers
- **Command:** stylesheet
- **Format:** PREFIX/das/DSN/stylesheet
- **Arguments:** None.
- **Return Document:**

```xml
<?xml version="1.0" standalone="no"?>
<!DOCTYPE DASSTYLE SYSTEM "dasstyle.dtd">
<DASSTYLE>
  <STYLELIST version="X.XX">
    <CATEGORY id="default">
      <TYPE id="default">
        <GLYPH> <ID>
          <ATTR>value</ATTR>
          <ATTR>value</ATTR>
          ...  
        </ID> </GLYPH>
      </TYPE>
    </CATEGORY>
    <CATEGORY id="category1">
      <TYPE id="default">
        <GLYPH> <ID>
          <ATTR>value</ATTR>
          <ATTR>value</ATTR>
        </ID> </GLYPH>
      </TYPE>
    </CATEGORY>
  </STYLELIST>
</DASSTYLE>
```

60
<TYPE id="type1">
  <GLYPH> <ID>
    <ATTR>value</ATTR>
  </ID> </GLYPH>
</TYPE>

...<CLASS> </CLASS>
</CATEGORY>

<CATEGORY id="category2">
  ...
</CATEGORY>

...</STYLE>
</STYLESHEET>
</DASSTYLE>

<!DOCTYPE> (required; one only) The doctype indicates which formal DTD specification to use. For the stylesheet query, the doctype DTD is "dasstyle.dtd".

<DASSTYLE> (required; one only) The appropriate doctype and root tag is DASSTYLE.

<STYLESHEET> (required; one only) There is a single <STYLE> tag. Its version (required) attribute indicates the current version of the stylesheet, and can be used for caching purposes.

<CATEGORY> (required; one or more) There are one or more <CATEGORY> tags, each providing information on the display of a high-level feature category. The id (required) tag uniquely names the category. A special name is “default”, which tells the annotation viewer what format to use for categories that are not otherwise specified in the stylesheet.

<TYPE> (required; one or more per CATEGORY) There are one or more <TYPE> tags per <CATEGORY>, each providing display suggestions for one type of annotation. The id (required) uniquely identifies the type. A special id is “default”, which, if present, identifies a default style for the enclosing category.

<GLYPH> (required; one per TYPE) There is a single <GLYPH> tag per <TYPE>. It provides information on what glyph (graphical widget) to use to display the indicated annotation type.
<ID> (required; one per GLYPH) The ID value refers to a recognized
glyph from the glyph types list (see Table A.1).

<ATTR> (optional; one or more per ID) The recognized ATTR
(attributes) are determined by which glyph ID is specified. See Table A.2
below for more information.

A.2 Feature Types and Categories

Features are associated with a graphical representation, a glyph, by the
type id and category. The "category" is designed to describe broad
annotation types (i.e. transcribed). The "id" refers to a more specific
instance within its category (i.e. exon). An annotation provider is free to
use any categories and types that they feel are appropriate to their
annotations. If an annotation is described by a type id and category which
are not defined by the accompanying stylesheet, programmatic defaults are
used. Table A.4 is a list of generic feature categories and specific feature
types within them. It is used as the contents of the programmatic default
stylesheet within Geodesic.
<table>
<thead>
<tr>
<th>Glyph Id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOX</td>
<td>A rectangular box</td>
</tr>
<tr>
<td>TOO MANY</td>
<td>Too many features than can be shown. Recommended for use in consolidating sequence homology hits. The recommended visual presentation is a set of overlapping boxes.</td>
</tr>
<tr>
<td>ARROW</td>
<td>A arrow with an axis either orthogonal or parallel to the sequence map.</td>
</tr>
<tr>
<td>LINE</td>
<td>A line. Lines are equivalent to arrows with both the NORTHEAST and SOUTHWEST attributes set to “no”.</td>
</tr>
<tr>
<td>CONNECTOR</td>
<td>The preferred graphical representation is as a “V” shaped line (commonly used to denote connections between exons).</td>
</tr>
<tr>
<td>TEXT</td>
<td>A bit of text.</td>
</tr>
<tr>
<td>EX</td>
<td>“X” marks the spot. Commonly used for point mutations and other point-like features.</td>
</tr>
<tr>
<td>CROSS</td>
<td>A cross “+”. Commonly used for point mutations and other point-like features.</td>
</tr>
<tr>
<td>DOT</td>
<td>A small circle. Commonly used for point mutations and other point-like features.</td>
</tr>
<tr>
<td>TRIANGLE</td>
<td>A triangle. Commonly used for point mutations and other point-like features.</td>
</tr>
<tr>
<td>SPAN</td>
<td>A spanning region, the recommended representation is a horizontal line with vertical lines at each end.</td>
</tr>
</tbody>
</table>

Table A.1: Valid glyphs for use in the stylesheet. The <ID> tag within the stylesheet is replaced by a particular glyph name. For example a box is designated by <BOX>. 

63
## Glyph Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>'COLOR'</td>
<td>The interior color of the glyph.</td>
</tr>
<tr>
<td>ALL GYPSHES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HEIGHT</td>
<td>INT</td>
<td>The height of the glyph. The height is orthogonal to the axis that defines</td>
</tr>
<tr>
<td>ALL GYPSHES EXCEPT TEXT</td>
<td></td>
<td>the extent of the feature on the sequence map.</td>
</tr>
<tr>
<td>BOX, TOOMANY, AND TRIANGLE ONLY:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OUTLINECOLOR</td>
<td>'COLOR'</td>
<td>The color of the glyph outline.</td>
</tr>
<tr>
<td>LINEWIDTH</td>
<td>INT</td>
<td>Width of the glyph outline.</td>
</tr>
<tr>
<td>ARROWS ONLY:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>PARALLEL</td>
<td>'BOOL'</td>
<td>Arrows run either parallel or orthogonal to the sequence axis.</td>
</tr>
<tr>
<td>NORTHEAST</td>
<td>'BOOL'</td>
<td>Arrow head is to the right (east) if the arrow runs parallel to the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and is up away from the sequence axis if the arrow runs orthogonal.</td>
</tr>
<tr>
<td>SOUTHWEST</td>
<td>'BOOL'</td>
<td>Arrow head is to the left (west) if the arrow runs parallel to the axis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>and is down towards the sequence axis if the arrow runs orthogonal.</td>
</tr>
<tr>
<td>TEXT ONLY:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FONT</td>
<td>'FONT'</td>
<td>The font.</td>
</tr>
<tr>
<td>FONTSIZE</td>
<td>INT</td>
<td>The font size.</td>
</tr>
<tr>
<td>STRING</td>
<td>STRING</td>
<td>The text to render.</td>
</tr>
<tr>
<td>STYLE</td>
<td>'FONT_STYLE'</td>
<td>The style in which to render this glyph. Multiple FONT_STYLE attributes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>may be present.</td>
</tr>
</tbody>
</table>

Table A.2: Each glyph attribute may take a particular value type. 1Colors can be specified using the “#RRGGBB” format commonly used in HTML or as one of the 16 IBM VGA colors recognized by Netscape and Internet Explorer. 2Boolean values are either "yes" or "no". 3Fonts can be any of the font identifiers recognized by Web browsers is acceptable, e.g. "helvetica". 4Font style is one of "bold", "italic", or "underline".
### Default Stylesheet Categories

<table>
<thead>
<tr>
<th>Category</th>
<th>Types</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translated</td>
<td>StopSite, ATG, CDS, 5'UTR, 3'UTR, Misc.translated</td>
</tr>
<tr>
<td>Transcribed</td>
<td>Exon, Intron, tRNA, mRNA, ncRNA, 5'Cap, PolyA, Splice5, Splice3, Misc_transcribed</td>
</tr>
<tr>
<td>Variation</td>
<td>Insertion, Deletion, Substitution, Misc_variation</td>
</tr>
<tr>
<td>Structural</td>
<td>Clone, Primer_left, Primer_right, Oligo, Misc_structural</td>
</tr>
<tr>
<td>Homology</td>
<td>NN (nucleotide to nucleotide), NP (nucleotide to peptide), PN (peptide to nucleotide), PP (peptide to peptide), Misc_homology</td>
</tr>
<tr>
<td>Repeat</td>
<td>Microsatellite, Inverted_repeat, Tandem_repeat, Transposable_element, LINE, Misc_repeat</td>
</tr>
<tr>
<td>Experimental</td>
<td>Knockout, Expression_tag, Microarrayed, RNAi, Transgenic, Mutant, Misc_experimental</td>
</tr>
</tbody>
</table>

Table A.4: A annotation provider is free to use any category and type name, but can expect the Geodesic client to have the above category-type pairs programmatically defined. Homologies should have a `<METHOD>` tag that indicates the algorithm used for the sequence comparison, and a `<TARGET>` tag in the `<GROUP>` field that indicates the target of the match. It is recommended, but not required, that the `<GROUP>` section contain `<LINK>` and/or `<NOTE>` tags that provide further information on the all features.
Appendix B

ACeDB Server HOWTO

The following is a description of how to setup an ACeDB based DAS server. This HOWTO was developed in collaboration with Robert Citrek in the Brent lab. See Appendix F.2 for more information on the various components. (It is assumed that make, gcc, and cvs are correctly installed.)

B.1 The Database

At http://www.acedb.org/Software/Downloads/ obtain the ACeDB software package. The current linux package is bin.linux.libc6.4.7l.tar.Z. It contains the programs and files necessary to create an ACeDB database. See the ACeDB documentation library, http://genome.cornell.edu/acedoc/, for more information on ACeDB. In the minimum, it will be necessary to learn how to load data into an ACeDB database. See Loading data into ACEDB by Dave Matthews and Suzanna Lewis and Tace by John Morris, both available from the ACeDB documentation library.

In addition, the ACeDB server package is needed. The current linux package is acedb_server_4.7l_linux_libc6.tar.gz which contains the gifaceserver. See the ACeDB server installation file, ftp://ftp.sanger.ac.uk/pub/acedb/SUPPORTED/SERVER.INSTALLATION for details on installation and setup. It is recommended to use aceclient, a client version of tace, to test the gifaceserver installation.
B.2 The Translator Script

The translator script is a Perl cgi script which requires AcePerl v1.65 or better. It is therefore necessary to have a server, for example Apache, which permits cgi script execution. Because the translator script communicates with the database through AcePerl, the server need not be on the same machine as the database. The mod_perl apache module P.2 is recommended, but not required.

Install Perl 5.005 or better and Digest::MD5, a perl module required by AcePerl. Both are available from CPAN (http://www.cpan.org/). Then install AcePerl using the RPC option (necessary with ACEDB 4.71).

The translator script, available in the CVS at /das/server/das, should be placed in a cgi executable location of the server. The das script's should be configured to reflect the location of the previously setup databases. The %DSN line contains:

- a DSN, the database identifier
- a description
- a dbserver, the name of the machine running gifaceserver
- a dbport, the port number used by gifaceserver
- a mapmaster, the URL of the database which is being annotated (where a reference database annotates itself)

The current translator script uses a hack to produce the stylesheet. The server's stylesheet should be placed in the same directory as the translator script in a file called style.xml. An example stylesheet is included in the CVS at /das/data/style.xml.

B.3 Testing

The setup of the server can be tested using any web browser. For example using lynx:
• "lynx -dump http://apache.machine.name/cgi-bin/das"
• "lynx -dump http://apache.machine.name/cgi-bin/das/dsn"
• "lynx -dump
  http://apache.machine.name/cgi-bin/das/DSN/entry_points"

The das query should return a usage statement. The dsn query should return an XML document generated from the %DSN table. The dsn query does not access the database. The final test above, entry_points, should contact the database and return the appropriate XML document.
Appendix C

Geodesic Documentation

C.1 How to obtain code

The source code for Geodesic is maintained under CVS control (See Appendix E.3). See http://biodas.org/geodesic/ for the latest news and status report on Geodesic.

The ISMB prototype version (v0.2) can be obtained from the CVS server using the tag “ismb00-v0.2.1”. Version 0.3 is the current version (under development) and requires no tag.

C.2 Installation and Setup Guide

Geodesic v0.2

Geodesic v0.2 requires Java 1.1 (See Appendix F.3) and the Neomorphic Genome Software Development Kit (NGSDK) (See Appendix F.3). It is recommended that the ngsdk.jar file be included in your CLASSPATH variable.

An example valid CLASSPATH (for DAS at /home/robin):
setenv CLASSPATH /home/robin/das/neomorphic/ngsdk.jar:/home/robin/das/::
In addition, the preferences file (das/data/pref.xml) must be copied into the main das directory (ex: das > cp data/pref.xml geodesic/pref.xml)

Compiling Source Code

change to the das/geodesic directory:
"cd das/geodesic/"

copy necessary xml file(s) from the data directory:
"cp ../data/pref.xml ."

compile (assumes correct CLASSPATH):
"geodesic> javac das.java"

execute:
"geodesic> java das"

Current Geodesic Version

Geodesic v0.3 and better requires Java 1.2 or greater (See Appendix F.3).

An example valid classpath (for das at /home/robin):
setenv CLASSPATH /home/robin/das/geodesic/:

In addition, the preferences file (das/data/preferences.xml) and bookmarks file (das/data/bookmarks.xml) must be copied into the main das directory (ex: das > cp data/preference.xml geodesic/preference.xml).

Compiling Source Code

change to the das/geodesic directory:
"cd das/geodesic/"

copy necessary xml file(s) from the data directory:
"cp ../data/preferences.xml ."
"cp ../data/bookmarks.xml ."
compile (assumes correct CLASSPATH):
"geodesic> javac Geodesic.java"

execute:
"geodesic> java Geodesic"
Appendix D

Geodesic Walk-Through

When using Geodesic, the initial behavior depends in part on the settings defined in the preferences.xml file. This walk-through shows how to use Geodesic’s menus and dialogs to select sources and load the clone ZK154. It should be noted that the CVS default preferences file will start Geodesic with the test servers pre-selected and the display centered on the clone ZK154.

The first thing a user must do is select a genome. The reference server selection dialog is under the Windows menu as “Change Genome”. Once selected, the Genome dialog box opens as shown in Figure D.1. The contents of the dialog box are those reference servers Geodesic has seen before. They are stored in the bookmarks.xml file. If the desired reference server is not available within the dialog, a URL can be entered. The genome dialog will take the URL, produce the appropriate “dsn” query, and add all discovered reference servers to the choices. In our case we select the Laptop C. elegans genome.

Once a reference genome is selected, the user can browse the genome using the Entry Points dialog, shown in Figure D.2. This is available from the Windows menu as “Entry Points”. When the user double clicks on a given entry point, the server is queried for that entry point’s substructure. When the user has found an entry point of interest, it is highlighted (by a single click) and then the “Display Feature” button will inform Geodesic that the main display should be focused on the selected entry point. Alternatively, if the user knows an entry point a priori, then they would use
Figure D.1: The genome dialog box allows the user to select a reference genome. The known genomes are stored within the bookmarks.xml file.
the Region dialog, shown in Figure D.3. For this walk-through, we use the region dialog to indicate that we'd like to view the clone ZK154.

Now we'd like to look at the annotations provided by the third party test servers. From the Windows menu choose “Annotations” to open the Select Annotations dialog box, as seen D.4. If third party sources are already shown in the display, then they are listed in the upper box. Those sources Geodesic knows about, from the bookmarks.xml file, which are compatible with the selected reference server (genome) are listed in the lower box. As with the Genome dialog, the user can enter a URL if the desired source is not available within the list. We have selected the tRNAscan-SE and the Exon/Intron (ORF) databases.

After a brief delay for a network retrieval, our display now looks like Figure D.5. In this display the data is split by strand, with forward information displayed above the axis and reverse strand information displayed below the axis. The stylesheet was loaded for each database. The tRNA database is using red boxes to show tRNAs. The ORF database is using blue boxes for exons, and blue connectors for introns. All three databases are using black boxes for their default. It can clearly be seen in the lorfdb layer that introns and exons are not yet properly grouped. The display preferences, can be adjusted within the Preferences menu option.

Double clicking on a feature brings up the “Feature Details” dialog, as shown in Figure D.6. This dialog is also available from the Windows menu. The information displayed is the basic contents of the DAS feature. When “link” queries work, it is from this dialog that users will be able to request more information from the providing server about this feature.

We would now like to save the DNA for this feature. With the feature still selected, we go to the File menu and select “Export” as “FASTA” for the “Selected Segment”. Geodesic will then query the reference server to retrieve the DNA segment corresponding to the feature. The resulting FASTA export dialog is shown in Figure D.7. The user may also choose to export features as tab delimited GFF or DAS XML. Alternatively, the entire contents of a source can be exported. All of these options are available within the File menu's Export option.
Figure D.2: The entry points dialog uses a file type structure display the structure of the reference server's map. It is anticipated that the DasMap graphical object can be adapted to present a graphical version of the entry points dialog in the near future.
Figure D.3: The region dialog box allows a user to enter an entry point directly.
Figure D.4: The annotations dialog box shows which annotation sources are currently selected as well as what other annotation sources are available. The mapmaster tag of each annotation source must match the currently selected genome to be visible within the annotation dialog box, see the DAS specification for more information about the mapmaster tag.
Figure D.5: A screen-shot of the current development version of Geodesic.
The view is on clone ZK154 in the *C. elegans* test server set.
Figure D.6: Example of feature details available for the selected annotation.
Figure D.7: Example of a feature being exported as a FASTA file. The coordinates of the feature are used to fetch the DNA from the reference server. The resulting FASTA file can be optionally saved to disk.
Appendix E

Project Support

E.1 Web site

The DAS project web site is http://biodas.org/ (formerly http://das.wustl.edu/). Because the distributed annotation system is a project which is still evolving, the information contained within this document may not accurately reflect the current state of the project. It is advisable to refer to the web site for the most current information.

E.2 License

The DAS project components are distributed under the GNU public license (GPL). For more information concerning the license terms, see the LICENSE and COPYING files available within the CVS or at the project web site.

E.3 CVS server

A CVS repository has been established for the DAS project. Anonymous (read only) access is available.

Accessing the repository:
cvs -d :pserver:anonymous@skynet.wustl.edu:/repository/das login
[give "anonymous" as the CVS password]
cvs -d :pserver:anonymous@skynet.wustl.edu:/repository/das checkout <module>
[creates the das/ directory]
cvs -d :pserver:anonymous@skynet.wustl.edu:/repository/das logout

Valid modules are:

das The entire project. Which includes the DAS specification, the
Wormbase ACeDB server, the source code for both versions of
Geodesic, a simple Java-based DAS specification validator, and the
biodas.org web site.

dasweb The contents of the biodas.org web site.

geodesic-0.2 The BOSC/ISMB version of Geodesic (v0.2) This version
uses AWT and is Neomorphic dependent.
geodesic The current version of Geodesic. This version uses Swing and is
Neomorphic independent.

E.4 Mailing List

A mailing list for DAS has been created by the EBI. To subscribe, send
mail to majordomo@alpha1.ybi.ac.uk with the command “subscribe das” in
the body (not the subject line) of the mail message.
Appendix F

Third Party Components and Technologies

F.1 Specification

**GFF** The general feature format (GFF) is a format for describing genes and other features associated with DNA, RNA and Protein sequences. The GFF format was designed to be easy to parse and process by a variety of programs in different languages. According to the GFF specification, “GFF is intentionally aiming for a low common denominator” between database systems. See [http://www.sanger.ac.uk/Software/formats/GFF/](http://www.sanger.ac.uk/Software/formats/GFF/) for more information.

**XML** The Extensible Mark-up Language (XML) is the universal format for structured documents and data on the Web. XML is a method for putting structured data in a text file. XML is license-free, platform-independent and well-supported. See [http://www.w3.org/XML/](http://www.w3.org/XML/) for more information.

**HTTP** Client queries are formatted URL requests which must follow the conventions of HTTP/1.0 protocol (RFC2616). See [http://www.w3.org/Protocols/](http://www.w3.org/Protocols/) for more information.
HTML  The web page returned from a links query must be valid HTML, compatible with current browsers. Ideally, a DAS client would turn over the display of HTML documents to the local browser. See http://www.w3.org/MarkUp/ for more information on HTML.

F.2 Server

Perl  The server's cgi translation script is written in Perl 5.005. See http://www.cpan.org/ for more information regarding Perl.

Apache  Apache is a powerful, flexible, HTTP/1.1 compliant web server which implements the latest protocols, including HTTP/1.1 and HTTP/1.0 (RFC2616). The DAS prototype system uses Apache 1.3.12 to respond to URL queries. See http://www.apache.org/ for more information on Apache.

mod_perl  With mod_perl it is possible to write Apache modules entirely in Perl. In addition, the persistent interpreter embedded in the server avoids the overhead of starting an external interpreter and the penalty of Perl start-up time. Consequently, while it is not required that mod_perl be installed, it is highly recommended for use with the DAS prototype server script. See http://perl.apache.org/ for more information on mod_perl.

AcePerl  AcePerl is an object-oriented Perl interface for the ACeDB database. It provides functionality for connecting to remote ACeDB databases, performing queries, fetching ACE objects, and database updating. It is required by the server cgi script to interface with the database's gifaceserver. For the DAS prototype system, AcePerl 1.65 or better is required. See http://stein.cshl.org/AcePerl/ for more information on AcePerl.

AceBrowser  AceBrowser is a small set of CGI scripts, written using AcePerl, which provide a straightforward browsable interface to an ACeDB database. AceBrowser was modified to produce the ACeDB dependent prototype system. It is not used in subsequent DAS prototype systems. See
http://stein.cshl.org/AcePerl/AceBrowser/ for more information on AceBrowser.

ACeDB  ACeDB is a genome database system developed in 1989 primarily by Jean Thierry-Mieg (CNRS, Montpellier) and Richard Durbin (Sanger Centre). ACeDB was originally developed for the C.elegans genome project, from which its name was derived (A C. elegans DataBase). However, the ACeDB tools have been generalized to be much more flexible and the same software is now used for many different genomic databases.

For the DAS prototype system, ACeDB is used as the database kernel which is access through the ACeDB server program called gifaceserver. The text interface program, Tace, is utilized for database construction and maintenance. AcePerl requires version 4.7l ("ell") or better. See http://www.acedb.org/ for more information on ACeDB.

F.3 Client

Java  The graphical client, Geodesic, was written in Java. The initial version, v0.2, used Java 1.1.3. Subsequent developments (v0.3 and beyond) utilized Java 1.2.2. See http://java.sun.com/ for more information on Java.

Ælfred  Ælfred is a fast, small, DTD-aware, event based, Java-based XML parser. It is designed to be extremely small and require very little memory to run. It is free for both commercial and non-commercial use and redistribution, provided that Microstar's copyright and disclaimer retain intact. Modification of Ælfred is also permitted provided that the modifications are clearly documented. See http://www.opentext.com/ for more information on the Ælfred parser.

Ælfred version 1.1 was utilized for all Geodesic XML parsing. It was modified to not utilize DTDs, increasing its speed and reducing its network requirements. A handler object was written to interface between the Ælfred parser and the data structures of Geodesic.
Neomorphic  The Neomorphic Genome Software Development Kit (NGSDK) is a set of Java-based components for building sophisticated bioinformatics data-visualization applications. It is an AWT compatible graphical library for use in Java 1.1 software developments. Geodesic v0.2 was developed using the NGSDK release v1.1.3 (Dec 1999). Under the conditions of the Neomorphic license, no redistribution is permitted. The package was obtainable from http://www.neomorphic.com/das/ but has not been available since Affymetrix bought Neomorphic in October 2000. Later versions of Geodesic (v0.3 or better) do not require the Neomorphic toolkit.
Bibliography


90


