

CONF-981113--

A Multi-User, Interactive, Annotated Flow-Chart Applet^{1,2}

S. E. Attenberger
Computational Physics and Engineering Division
Oak Ridge National Laboratory
P.O. Box 2008
Oak Ridge, Tennessee 37831-6418

+1 423 241 5929
sea@ornl.gov

Category of Submission: Paper
Primary Contact: S. E. Attenberger

RECEIVED

MAY 14 1998

OSTI

¹Research sponsored by the National Aeronautics and Space Administration under Interagency Agreement DOE No. 2013-F044-A1 under Lockheed Martin Energy Research Corp., contract DE-AC05-84OR21400 with the U.S. Department of Energy.

"The submitted manuscript has been authorized by a contractor of the U.S. Government under contract No. DE-AC05-96OR22464. Accordingly, the U.S. Government retains a non-exclusive, royalty-free license to publish or reproduce the published form of this contribution, or allow others to do so, for U.S. Government purposes."

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

MASTER

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

A Multi-User, Interactive, Annotated Flow-Chart Applet

S. E. Attenberger

Oak Ridge National Laboratory, Oak Ridge, TN USA

+1 423 241 5929

sea@ornl.gov

ABSTRACT

This paper describes a web-based documentation tool that has been developed for use by a team of collaborators who are developing interacting components within a system. It consists of a Java applet in the form of a flow chart with additional text that can be viewed by clicking on the desired component. The text, positions, and linkages of the components may be edited by the user and saved to the server. The concept could easily be applied to other collaborative work. Using the techniques and code segments described here, a programmer should be able to customize a similar applet for his own use in a few days time.

Keywords

Java, perl, applet, flow chart, annotate, multi-user

INTRODUCTION

This paper describes a documentation tool recently developed by the Oak Ridge National Laboratory (ORNL) Distributed Active Archive Center (DAAC), a NASA-funded institution for archiving biogeochemical dynamics data. The DAAC accepts data from principal investigators and distributes it to modelers and to the general public all over the world. The data are accessed through a constantly evolving complex of hardware and software that is maintained by the collaborative effort of a team of systems personnel.

The initial problem was to communicate the existing system configuration to some new collaborators. A static (Visio) flow chart document was used at first, but as various collaborators contributed to the document it became inadequate to display all of the relevant information. Furthermore the system configuration continued to evolve as the document was developed, and the document was perpetually outdated. There was a clear need for a documentation tool that could be modified easily by whichever collaborator changed a part of the configuration. To meet this need, a Java applet was developed that permits systems personnel to document modifications by using a

shared online document. The software aims to provide a comprehensible overview of all the major components, with additional information available by clicking on components of interest.

We begin with an example of a typical configuration diagram. Then we show how to use the applet to construct a configuration diagram. Next we describe the essential features of the implementation, including segments of Perl and Java code. Finally we discuss future directions and plans.

EXAMPLE: A TYPICAL CONFIGURATION DIAGRAM

Fig. 1 shows an overview of the DAAC system configuration that has been simplified for this presentation. In actual use the screen would contain names of directories, files, and database tables. The URLs labeled "Public" are real and the general public is encouraged to view our web site, however this applet runs on an internal server that is not available to the general public.

The core of the configuration is the actual data, which is stored primarily as files on spinning disks. A Sybase database stores metadata which describes the data using standard keywords. Multiple local and remote search engines present this metadata to the user to help him or her find relevant data.

The search engines are represented here as the peripheral boxes in the diagram. For simplicity we have lumped the search engines together with the corresponding URLs and servers. There are actually 8 DAACs in the U.S., and each of them have a V0 search engine that can search the V0 servers of all 8 sites, although only 2 of them are shown in this diagram. The V0 system allows the user to arrange for data delivery via ftp and other means. Most users prefer the

1. Research sponsored by the National Aeronautics and Space Administration under Interagency Agreement DOE No. 2013-F044-A1 under Lockheed Martin Energy Research Corp., contract DE-AC05-84OR21400 with the U.S. Department of Energy.

2. Version 0 of the Information Management System (IMS) of the Earth Observing System Data Information System (EOSDIS)

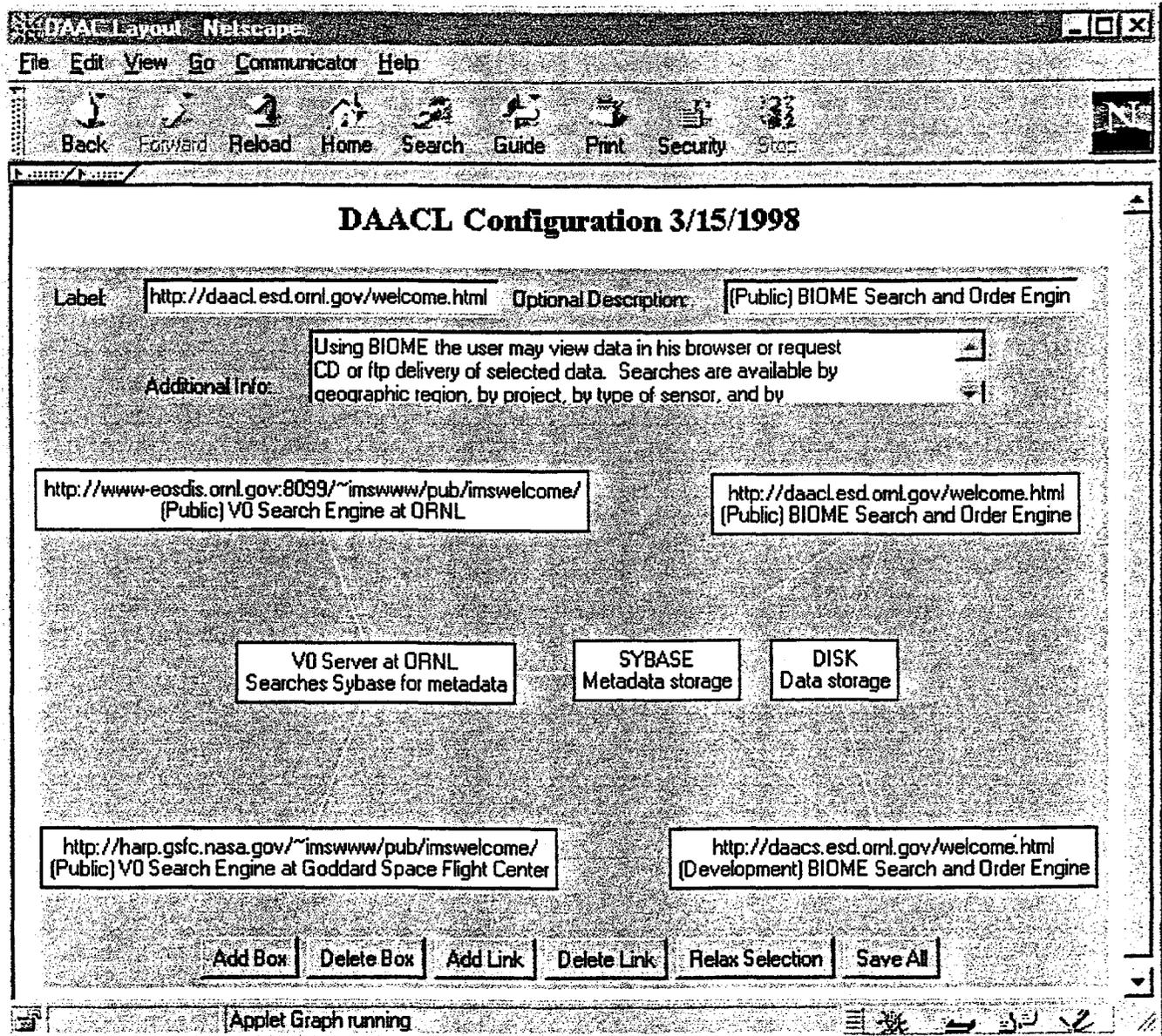


Fig. 1 A Typical Configuration Diagram

BIOME search engine even though it does not connect to the other DAACs because it gives the user the option to view the data immediately in their browser.

In the figure the "http://daacl.esd.ornl.gov/welcome.html" box has been selected by clicking on it, which causes the Java text boxes at the top of the applet to display the corresponding fields for title, description, and "Additional Info". Although it doesn't show in this black and white reproduction, the color of the box changes upon selection. The text in any of the three text boxes may be edited.

The lines joining the boxes are called "links" (not to be confused with hyperlinks) and they indicate flow of information. Usually the flow is bidirectional so no

arrowhead option has been implemented in this version.

CONSTRUCTING A CONFIGURATION DIAGRAM

Fig.2 shows an early stage in the construction of the previously described configuration. To add a box representing a Node object, one clicks the "Add Box" button. A box appears labeled "New Box", with empty Description and Additional Info fields. The fields can be edited immediately or later, by clicking on the desired box to select it. As the user fills the text boxes, the change appears simultaneously in the figure. In the present version links are added by clicking 2 boxes and then clicking "Add Link", but in the future a more intuitive rubber band approach will be used.

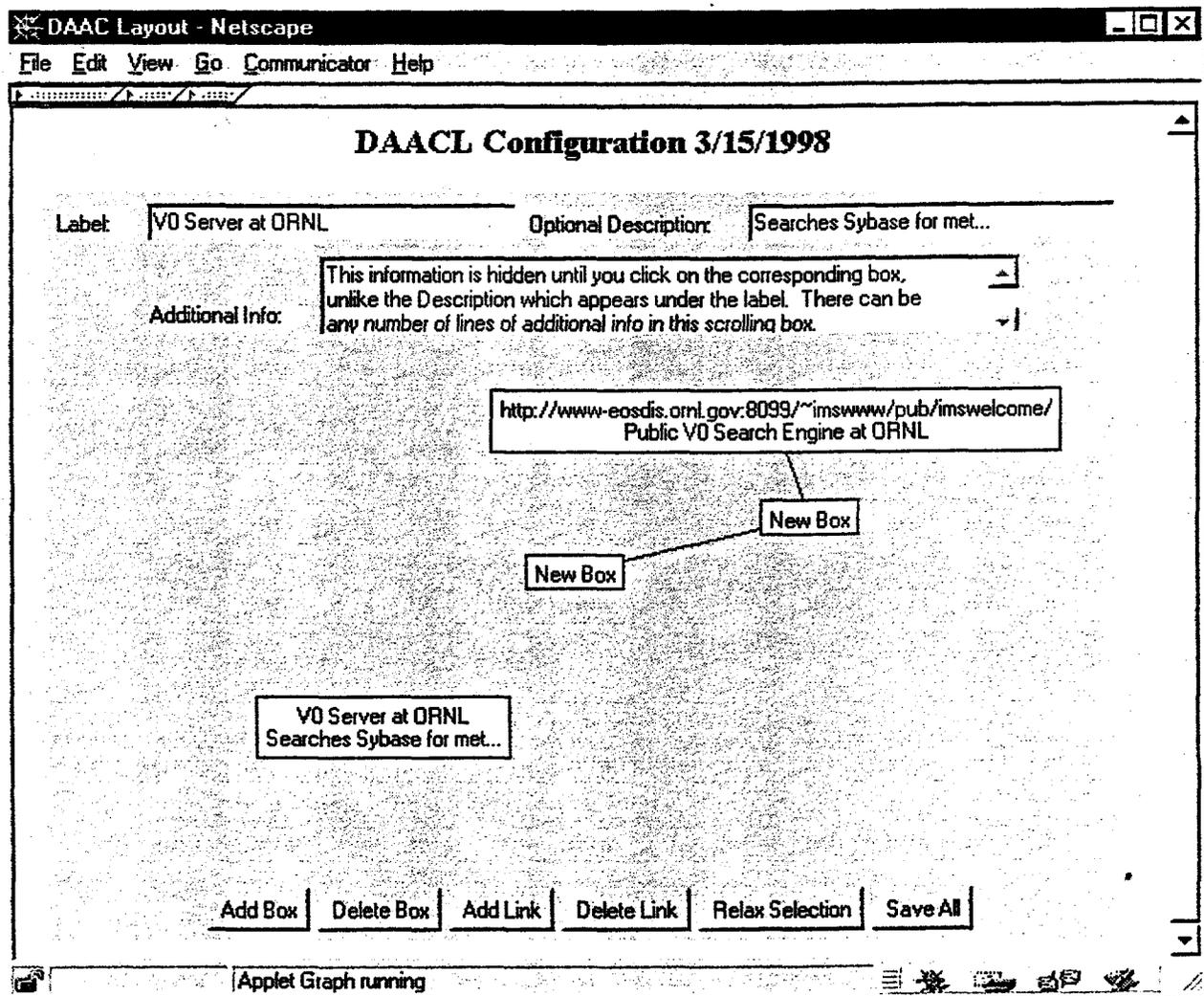


Fig. 2 Using the applet to construct a configuration diagram.

We have experimented with an algorithm to let the applet choose optimal placement of the boxes (using the "Relax Selection" button), but to date this has not been very successful. We have some ideas for improvements that will be discussed in the "Future Directions and Plans" section. The user may at any point click on "Save All" in order to save the modified configuration in disk files on the server.

IMPLEMENTATION

The application uses HTML, perl, and Java files that

HTML File for Initial Entry

```
<html><header>
<title> DAAC Layout </title>
```

reside on a unix server. The initial invocation is to an HTML file with an embedded Server-Side Include that merely calls a perl script. The perl (CGI) script writes the HTML dynamically to the browser, the applet being invoked within the HTML. Thereafter, the perl script gets called directly by the applet when the user clicks on "Save All", with all information about the configuration being passed in a query string. The HTML file is used only because the applet sits in the HTML disk area rather than the CGI disk area. This causes Java to look for the applet in the proper place.

```

</header> <body>
<!--#exec cgi="/cgirootdirectory/config2.p"-->
</body> </html>

```

Segments of CGI script "config2.p"

```

require      "cgi-lib.pl";
  &ReadParse(*input);
  $linkString=$input{'linkString'};
  $nodeString=$input{'nodeString'};

#Save the configuration to disk if it has been initialized
  if ($linkString ne "") {
    open(LINKS,">$LinksFile");print LINKS "$linkString";
    close LINKS;
  }

  if ($nodeString ne "") {
    open(NODES,">$NodesFile");print NODES "$nodeString";
    close NODES;
  }

print "Content-type: text/html\n\n";
#On the first pass we must write the html containing the applet.
#On later passes, it does no harm...
  print <<"EOF".
<html><header>
<title> DAAC Layout </title>
</header>
<BODY bgcolor="#FFFFFF">
<center><H3>DAACL Configuration</H3></center>

<applet codebase="http://developmentarea/" code="Graph.class" width=600 height=400>
<param name=links value="
EOF
;

  open(LINKS,"$LinksFile"); while (<LINKS>) {print;}
  close LINKS;

  print <<"EOF"
">
<param name=fixed value="
EOF
;

  open(NODES,"$NodesFile"); while (<NODES>) {print;}
  close NODES;

  print <<"EOF"
">
</applet></body> </html>
EOF
;

```

Segments of the Applet

```

public class Graph extends Applet {

```

```

boolean saveAll() {
    String URLstr = RootDir + "/config2.p?nodeString=";
    Dimension d = size();
    //Format for nodes.dat
    //index|xc|yc|label|description|hiddenText|
    Enumeration eNode = panel.nodes.elements();
    while (eNode.hasMoreElements()) {
        Node n = (Node)eNode.nextElement();
        URLstr = URLstr + n.index + "|";
        int xp = (int)(Math.round(100 * (n.xc / d.width)));
        int yp = (int)(Math.round(100 * (n.yc / d.height)));
        URLstr = URLstr + xp + "|" + yp + "|";
        URLstr = URLstr + noSpace(n.lbl) + "|";
        URLstr = URLstr + noSpace(n.descr) + "|";
        URLstr = URLstr + noSpace(n.hiddenText) + "|";
    }
    URLstr = URLstr+"&linkString=";
    Enumeration eLink = panel.links.elements();
    while (eLink.hasMoreElements()) {
        Link el = (Link)eLink.nextElement();
        URLstr = URLstr + el.from.index + ">";
        URLstr = URLstr + el.to.index + "|";
    }
    URL theURL = null;
    try { theURL = new URL(URLstr); }
    catch (MalformedURLException ex) {
        System.out.println("Bad URL: " + theURL);
        return false;
    }
    getAppletContext().showDocument(theURL);
    return true;
} //saveAll()

```

FUTURE DIRECTIONS AND PLANS

The complexity of the DAAC configuration is a challenge to represent graphically, and we continue to search for ways to clarify and condense the representation. It is helpful that we typically view the applet on very large screens, allowing much more detail than is shown in Fig.1. In the time between this writing and the 1998 CSCW meeting, there should be much experience gained in using the applet and many features will be added or improved.

In the immediate future we plan to lock the configuration data files when someone checks them out for modification. This will avoid interference between simultaneous modifications by different users. It would be natural to combine this with a log of modifications, including dates and user Ids.

It would be nice to be able to input an arbitrarily linked complex of nodes and have the applet automatically arrange them in an optimal order. We have attempted variations that mimic molecular dynamics, but so far we always revert to positioning everything by hand. We believe that it may help to separate those nodes that have many links and to have simply linked boxes tend to migrate toward the edges. At present the links pass under each other and under boxes, which is not very satisfactory. Also the links always originate and end at box centers, which is unnecessarily restrictive. An algorithm to reduce the line crossings by using random perturbations and/or Jacobean techniques, is probably needed. After the configuration is optimized, we might attempt a Visio-like linkage where the links bend around the boxes.

We expect that the applet, which is already useful in its initial version, will become even more useful and user-friendly as we continue to enhance it.